Naval Research Laboratory
Monterey, CA 93943-5502

NRL/MR/7532--93-7212

The NOGAPS Ten Year AMIP Integration

THOMAS E. ROSMOND

Prediction Systems Branch
Marine Meteorology Division

October 1993

Approved for public release; distribution unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. Agency Use Only *(Leave blank).*	2. Report Date.	3. Report Type and Dates Covered.
	October 1993	Final

4. Title and Subtitle.
The NOGAPS Ten Year AMIP Integration

5. Funding Numbers.
PE 62704N
PN 7W0513
AN DN659751

6. Author(s).
Thomas E. Rosmond

7. Performing Organization Name(s) and Address(es).
Naval Research Laboratory, Marine Meteorology Division
7 Grace Hopper Avenue stop 2
Monterey, CA 93943-5502

8. Performing Organization Report Number.
NRL/MR/7532--93-7212

9. Sponsoring/Monitoring Agency Name(s) and Address(es).
Office of Naval Research
Arlington, VA 22217-5660

10. Sponsoring/Monitoring Agency Report Number.

11. Supplementary Notes.

12a. Distribution/Availability Statement.
Approved for public release; distribution unlimited.

12b. Distribution Code.

13. Abstract *(Maximum 200 words).*

This report is a user's guide to the 10 year model history data set from an integration of the Navy Operational Global Atmospheric Prediction System (NOGAPS). NRL Monterey is participating in the Atmospheric Model Intercomparison Project (AMIP) sponsored by the Department of Energy. Nearly every major atmospheric modeling center in the world has run their model for the years 1979 through 1988 with sea surface temperature and sea ice boundary conditions provided by AMIP. The NOGAPS AMIP data set has great potential for use in a wide variety of atmosphere/ocean studies that require the high time and space resolution available from a global atmospheric model.

14. Subject Terms.
NOGAPS Climate modeling
Global modeling Air-sea interaction

15. Number of Pages.
72

16. Price Code.

17. Security Classification of Report.	18. Security Classification of This Page.	19. Security Classification of Abstract.	20. Limitation of Abstract.
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	Same as report

CONTENTS

THE NOGAPS TEN YEAR AMIP INTEGRATION

1. INTRODUCTION

Numerical global models of the atmosphere have reached a level of sophistication that makes them our most powerful research tool for studying large-scale atmospheric processes. These general circulation models (GCMs) are widely used for numerical weather prediction (NWP) and for climate simulation. Until recently NWP models had relatively crude physical parameterizations compared to climate models, but the almost universal adoption of global NWP models and increased emphasis on medium range prediction (5-10 days) has motivated improvement of NWP model physics. Now there is usually little difference between GCMs used for NWP and for climate studies except that operational NWP models are normally run with significantly higher resolution than are climate GCMs.

There are now dozens of GCMs being run at atmospheric research organizations and operational NWP centers around the world. The complexity of atmospheric processes has lead to a considerable lack of unanimity among atmospheric scientists about how to model these processes. Therefore there is great diversity among the GCMs' physical parameterizations. Such diversity is valuable as a path toward determining the "best" parameterizations, provided there is some kind of rigorous model intercomparison effort under controlled conditions.

The U.S Department of Energy's Lawrence Livermore National Laboratory (LLNL) has recognized the need for comprehensive GCM intercomparison and established the Program for Climate Model Diagnosis and Intercomparison (PCMDI). In cooperation with the World Climate Research Programme's Working Group on Numerical Experimentation (WGNE) PCMDI has established the Atmospheric Model Intercomparison Project (AMIP) as a way of bringing many of the world's GCMs together for a comprehensive intercomparison. The numerical experiment run by all participants is a 10 year integration over the years 1979-1988, with PCMDI providing prescribed sea surface temperature (SST) and sea ice coverage for this period. The solar constant and atmospheric CO2 concentration is also specified by PCMDI. All other model parameters and degrees of freedom such as resolution are at the discretion of the participants.

Participation by the world's atmospheric modeling groups in AMIP has been almost unanimous. AMIP is by far the largest numerical experiment of this kind ever attempted. In addition to basic research modeling groups, every major operational NWP center in the world is participating, clear evidence that these centers believe their models are legitimate climate GCMs and can be evaluated as such. PCMDI provides data management and other logistical support to the participating GCM groups, and is also supporting several groups with computer time on LLNL computer systems. PCMDI's ultimate goal is to collect the 10 year model output histories from each participating group. They are requesting each group to output histories every six hours of integration, so the volume of data to be handled is enormous; about 2 terabytes.

The Navy Research Laboratory (NRL) is participating in AMIP because the Navy has great interest in improved global atmospheric forecasts. The Navy Operational Global Atmospheric Prediction System (NOGAPS), developed over the past several years by Navy atmospheric modelers in Monterey, Ca., generates or directly influences a wide variety of meteorological and oceanographic operational products for Navy fleet users. NOGAPS improvements will directly benefit these users by improving every NOGAPS product. AMIP is an opportunity to compare NOGAPS to the other major GCMs around the world, showing strengths and weaknesses that can guide research on future model improvements. The 10 year AMIP NOGAPS history is also an invaluable data set supporting a wide variety of NRL research projects.

A summary of the NOGAPS forecast model features and some details of how the model was modified for the AMIP experiment are given in section 2 of this report. The large volume of model history data produced by the NOGAPS AMIP integration required special attention to data organization and storage. Section 3 describes the data management strategy used for the NOGAPS AMIP data set. In section 4 the model parameters available in the AMIP data set are itemized. Concluding comments and data access procedures are given in section 5. The appendix contains listings of Fortran software which is available to facilitate access to the AMIP data and post-processing of the raw history files.

2. NOGAPS AMIP MODEL DESCRIPTION

A detailed description of the NOGAPS forecast model is in Hogan et al. (1992). Table 1 summarizes the major features of the model. Briefly, the model is a global spectral model similar in concept to the operational global NWP models run by other large operational centers. The physical parameterizations are representative of those used in other NWP models and climate GCMs run by many major meteorological research centers. The version of NOGAPS run for the AMIP 10 year integration is very similar to the system run operationally by Fleet Numerical Oceanography Center (FNOC) in Monterey. There are no significant differences between the physical parameterizations of the operational NOGAPS and the AMIP model. The AMIP integration was done on the CRAY YMP at Stennis Space Center (SSC), Mississippi, so the floating point results are 64 bit precision, while the operational NOGAPS runs on the FNOC CYBER 205 using 32 bit operations. No meteorologically significant differences have been observed because of this word size difference, however. To keep computer time costs within reasonable bounds the AMIP NOGAPS has a horizontal resolution of triangular truncation wavenumber 47 (T47) rather than the T79 resolution of the operational NOGAPS. The number of levels is 18, the same as used operationally. Interestingly, T47 is a significantly higher resolution than most of the other participating AMIP models.

Table 1: Description of NOGAPS Forecast Model.

- SPECTRAL TRIANGULAR TRUNCATION

    - RESOLUTION
        CURRENT OPERATIONAL: T79/L18
        FUTURE: ~T150/L36

COMPUTATIONAL DETAILS

    - SEMI-IMPLICIT TIME DIFFERENCING

    - IMPLICIT ZONAL ADVECTION
        VORTICITY
        MOISTURE

    - SILHOUETTE OROGRAPHY

    - PLANETARY BOUNDARY LAYER
        EXPLICITLY RESOLVED
        STABILITY DEPENDENT MIXING COEFF.
        SHALLOW CUMULUS
        PREDICTED GROUND HYDROLOGY

    - RADIATION
        DIURNAL CYCLE
        PREDICTED STABLE/CUMULUS CLOUDS
        OZONE HEATING

    - CUMULUS
        ARAKAWA-SCHUBERT
        DOWNDRAFTS
        MASS FLUX KERNEL OR "RELAXED" SCHEMES

    - LARGE SCALE PRECIPITATION

    - GRAVITY WAVE DRAG

A mean solar constant of 1365 watts m^{-2} and a global mean CO2 concentration of 345 ppm was specified by PCMDI for all participating models. These are standard input parameters of the model and are quite close to the values used in the operational NOGAPS. The bottom boundary conditions of SST and sea ice required some model modification, however. When NOGAPS runs as an operational NWP model, the SST and sea ice is updated at the beginning of each forecast, but held fixed during typical NWP integrations of a week or less. For the ten year AMIP integration NOGAPS was modified to allow continuous updating of the SST and sea ice, as is done in other climate GCMs.

As stated above, PCMDI provided the SST and sea ice coverage to be used by all AMIP models. These are monthly mean 2.0 x 2.0 deg fields covering the 1979-1988 period and were prepared jointly by Climate Analysis Center (CAC) of NOAA and the Center for Ocean-Land-Atmosphere (COLA) Interactions at the University of Maryland. The monthly averages are assumed valid at the middle of each month. The SST and sea ice fields are bi-cubic spline interpolated to the approximately 2.5 deg Gaussian transform grid of the T47 NOGAPS and linearly interpolated in time every model time step to ensure smoothly varying bottom boundary conditions.

3. AMIP DATA MANAGEMENT STRATEGY

A numerical experiment on the scale of the 10 year NOGAPS AMIP integration requires careful consideration of the organization of the model history files to allow reasonably flexible access to the data. The NOGAPS AMIP history is about 100 Gbytes, so it presents a daunting challenge to anyone interested in exploiting this unique data set. Special software was written for the forecast model to allow easy model restarts and reruns as needed during the integration. This software is also useful for model history post-processing. The Fortran subroutines are listed in the appendix and are available on the SSC computer system.

To understand the data management philosophy used for the NOGAPS AMIP model integration history, a brief description of the SSC computer system is necessary. The heart of the system is an 8 processor, 128 Mword CRAY YMP with about 40 Gbytes of on-line disk storage. A second single processor YMP acts as a dedicated file server connected to the large machine via a 1.0 Gbit/sec network. The file server also has about 40 Gbytes of disk storage and is connected to a tape cartridge archive system with several Terabytes of capacity. Several other smaller computers and workstations are also on the high speed network. The important distinction between the two CRAY's is the way the disk storage is managed on the two systems. The big machine's disks are all defined as "scratch" space, with file duration of only a few hours or days, depending on file size and access frequency. Access to these scratch files is direct, however, so I/O performance is quite good.

The disks physically attached to the file server CRAY, on the other hand, are defined as a "shared" file system among all the machines on the high speed network, including the big CRAY. This means that jobs running on the big CRAY can functionally access files on this shared file system just like scratch files. There is a big performance penalty, however, since shared file data must be physically moved from the file server to the big machine across the high-speed network. A high-speed network, even with a 1.0 Gbit bandwidth, is much slower than direct access disks.

Files on the shared system are subject to migration to the archive system under the control of sophisticated UNICOS file management software, also based on file size and access frequency. Retrieval of files from the archive is "transparent" to users, except for noticeable time delays as the data moves from the archive tape cartridges back to the shared disks. During heavy system use periods these delays can be quite significant as network traffic overwhelms the ability of the system to efficiently move data.

The design of this multi-level file system structure dictated that some care be taken in running data intensive computations such as the NOGAPS AMIP integration. The model typically ran one month integration period, during which model history was written to "scratch" files. At the end of each month's integration, this history was copied to a shared file system directory using the UNIX "tar" utility. In the process about 500 history files were consolidated into 5 tar files and 2 small auxiliary files.

Table 2. Example of AMIP directory "pcmdi0885", showing 5 tar files and 2 small auxiliary files. The expanded contents of the tar files are shown in Table 3.

-rw-rw-r--	1	rosmfnoc	528482304	Sep	30	1992 c3ave0885.tar
-rw-rw-r--	1	rosmfnoc	124583936	Sep	30	1992 c3grid0885.tar
-rw-rw-r--	1	rosmfnoc	23724032	Sep	30	1992 c3spec0885.tar
-rw-rw-r--	1	rosmfnoc	8760	Sep	30	1992 hccn32
-rw-rw-r--	1	rosmfnoc	5373952	Sep	30	1992 lsifil0885.tar
-rw-rw-r--	1	rosmfnoc	172097536	Sep	30	1992 shist0885.tar
-rw-rw-r--	1	rosmfnoc	124464	Sep	30	1992 trpfil

The complete NOGAPS AMIP history data base is located on the SSC YMP8128 (designated LSC) in the master directory '/u/b/rosmond/pcmdi'. In this directory are subdirectories containing the 120 monthly histories from January 1979 through December 1988. The naming convention of the subdirectories is 'pcmdiMMYY', so for example 'pcmdi0885' contains the history for August 1985. Table 2 displays the contents of 'pcmdi0885' using the UNIX command 'ls -la'. Note that 'tar' files are identified with a '.tar' extension. The size of each file is given in bytes; all files for a month total about 800 megabytes. Table 3 shows, using the UNIX command 'ls', an example of the expanded contents of the tar files. Except for the two small

Table 3. Expanded contents of tar files in AMIP directory "pcmdi0885".

c3ave057696	c3grid057966	c3grid058428	c3spec058140	lsifil057846	lsifil058308	shist058020
c3ave057720	c3grid057972	c3grid058434	c3spec058146	lsifil057852	lsifil058314	shist058026
c3ave057744	c3grid057978	c3grid058440	c3spec058152	lsifil057858	lsifil058320	shist058032
c3ave057768	c3grid057984	c3spec057696	c3spec058158	lsifil057864	lsifil058326	shist058038
c3ave057792	c3grid057990	c3spec057702	c3spec058164	lsifil057870	lsifil058332	shist058044
c3ave057816	c3grid057996	c3spec057708	c3spec058170	lsifil057876	lsifil058338	shist058050
c3ave057840	c3grid058002	c3spec057714	c3spec058176	lsifil057882	lsifil058344	shist058056
c3ave057864	c3grid058008	c3spec057720	c3spec058182	lsifil057888	lsifil058350	shist058062
c3ave057888	c3grid058014	c3spec057726	c3spec058188	lsifil057894	lsifil058356	shist058068
c3ave057912	c3grid058020	c3spec057732	c3spec058194	lsifil057900	lsifil058362	shist058074
c3ave057936	c3grid058026	c3spec057738	c3spec058200	lsifil057906	lsifil058368	shist058080
c3ave057960	c3grid058032	c3spec057744	c3spec058206	lsifil057912	lsifil058374	shist058086
c3ave057984	c3grid058038	c3spec057750	c3spec058212	lsifil057918	lsifil058380	shist058092
c3ave058008	c3grid058044	c3spec057756	c3spec058218	lsifil057924	lsifil058386	shist058098
c3ave058032	c3grid058050	c3spec057762	c3spec058224	lsifil057930	lsifil058392	shist058104
c3ave058056	c3grid058056	c3spec057768	c3spec058230	lsifil057936	lsifil058398	shist058110
c3ave058080	c3grid058062	c3spec057774	c3spec058236	lsifil057942	lsifil058404	shist058116
c3ave058104	c3grid058068	c3spec057780	c3spec058242	lsifil057948	lsifil058410	shist058122
c3ave058128	c3grid058074	c3spec057786	c3spec058248	lsifil057954	lsifil058416	shist058128
c3ave058152	c3grid058080	c3spec057792	c3spec058254	lsifil057960	lsifil058422	shist058134
c3ave058176	c3grid058086	c3spec057798	c3spec058260	lsifil057966	lsifil058428	shist058140
c3ave058200	c3grid058092	c3spec057804	c3spec058266	lsifil057972	lsifil058434	shist058146
c3ave058224	c3grid058098	c3spec057810	c3spec058272	lsifil057978	lsifil058440	shist058152
c3ave058248	c3grid058104	c3spec057816	c3spec058278	lsifil057984	shist057696	shist058158
c3ave058272	c3grid058110	c3spec057822	c3spec058284	lsifil057990	shist057702	shist058164
c3ave058296	c3grid058116	c3spec057828	c3spec058290	lsifil057996	shist057708	shist058170
c3ave058320	c3grid058122	c3spec057834	c3spec058296	lsifil058002	shist057714	shist058176
c3ave058344	c3grid058128	c3spec057840	c3spec058302	lsifil058008	shist057720	shist058182
c3ave058368	c3grid058134	c3spec057846	c3spec058308	lsifil058014	shist057726	shist058188
c3ave058392	c3grid058140	c3spec057852	c3spec058314	lsifil058020	shist057732	shist058194
c3ave058416	c3grid058146	c3spec057858	c3spec058320	lsifil058026	shist057738	shist058200
c3ave058440	c3grid058152	c3spec057864	c3spec058326	lsifil058032	shist057744	shist058206
c3grid057696	c3grid058158	c3spec057870	c3spec058332	lsifil058038	shist057750	shist058212
c3grid057702	c3grid058164	c3spec057876	c3spec058338	lsifil058044	shist057756	shist058218
c3grid057708	c3grid058170	c3spec057882	c3spec058344	lsifil058050	shist057762	shist058224
c3grid057714	c3grid058176	c3spec057888	c3spec058350	lsifil058056	shist057768	shist058230
c3grid057720	c3grid058182	c3spec057894	c3spec058356	lsifil058062	shist057774	shist058236
c3grid057726	c3grid058188	c3spec057900	c3spec058362	lsifil058068	shist057780	shist058242
c3grid057732	c3grid058194	c3spec057906	c3spec058368	lsifil058074	shist057786	shist058248
c3grid057738	c3grid058200	c3spec057912	c3spec058374	lsifil058080	shist057792	shist058254
c3grid057744	c3grid058206	c3spec057918	c3spec058380	lsifil058086	shist057798	shist058260
c3grid057750	c3grid058212	c3spec057924	c3spec058386	lsifil058092	shist057804	shist058266
c3grid057756	c3grid058218	c3spec057930	c3spec058392	lsifil058098	shist057810	shist058272
c3grid057762	c3grid058224	c3spec057936	c3spec058398	lsifil058104	shist057816	shist058278
c3grid057768	c3grid058230	c3spec057942	c3spec058404	lsifil058110	shist057822	shist058284
c3grid057774	c3grid058236	c3spec057948	c3spec058410	lsifil058116	shist057828	shist058290
c3grid057780	c3grid058242	c3spec057954	c3spec058416	lsifil058122	shist057834	shist058296
c3grid057786	c3grid058248	c3spec057960	c3spec058422	lsifil058128	shist057840	shist058302
c3grid057792	c3grid058254	c3spec057966	c3spec058428	lsifil058134	shist057846	shist058308
c3grid057798	c3grid058260	c3spec057972	c3spec058434	lsifil058140	shist057852	shist058314
c3grid057804	c3grid058266	c3spec057978	c3spec058440	lsifil058146	shist057858	shist058320
c3grid057810	c3grid058272	c3spec057984	hccn32	lsifil058152	shist057864	shist058326
c3grid057816	c3grid058278	c3spec057990	lsifil057696	lsifil058158	shist057870	shist058332
c3grid057822	c3grid058284	c3spec057996	lsifil057702	lsifil058164	shist057876	shist058338
c3grid057828	c3grid058290	c3spec058002	lsifil057708	lsifil058170	shist057882	shist058344
c3grid057834	c3grid058296	c3spec058008	lsifil057714	lsifil058176	shist057888	shist058350
c3grid057840	c3grid058302	c3spec058014	lsifil057720	lsifil058182	shist057894	shist058356
c3grid057846	c3grid058308	c3spec058020	lsifil057726	lsifil058188	shist057900	shist058362
c3grid057852	c3grid058314	c3spec058026	lsifil057732	lsifil058194	shist057906	shist058368
c3grid057858	c3grid058320	c3spec058032	lsifil057738	lsifil058200	shist057912	shist058374
c3grid057864	c3grid058326	c3spec058038	lsifil057744	lsifil058206	shist057918	shist058380
c3grid057870	c3grid058332	c3spec058044	lsifil057750	lsifil058212	shist057924	shist058386
c3grid057876	c3grid058338	c3spec058050	lsifil057756	lsifil058218	shist057930	shist058392
c3grid057882	c3grid058344	c3spec058056	lsifil057762	lsifil058224	shist057936	shist058398
c3grid057888	c3grid058350	c3spec058062	lsifil057768	lsifil058230	shist057942	shist058404
c3grid057894	c3grid058356	c3spec058068	lsifil057774	lsifil058236	shist057948	shist058410
c3grid057900	c3grid058362	c3spec058074	lsifil057780	lsifil058242	shist057954	shist058416
c3grid057906	c3grid058368	c3spec058080	lsifil057786	lsifil058248	shist057960	shist058422
c3grid057912	c3grid058374	c3spec058086	lsifil057792	lsifil058254	shist057966	shist058428
c3grid057918	c3grid058380	c3spec058092	lsifil057798	lsifil058260	shist057972	shist058434
c3grid057924	c3grid058386	c3spec058098	lsifil057804	lsifil058266	shist057978	shist058440
c3grid057930	c3grid058392	c3spec058104	lsifil057810	lsifil058272	shist057984	trpfil
c3grid057936	c3grid058398	c3spec058110	lsifil057816	lsifil058278	shist057990	
c3grid057942	c3grid058404	c3spec058116	lsifil057822	lsifil058284	shist057996	
c3grid057948	c3grid058410	c3spec058122	lsifil057828	lsifil058290	shist058002	
c3grid057954	c3grid058416	c3spec058128	lsifil057834	lsifil058296	shist058008	
c3grid057960	c3grid058422	c3spec058134	lsifil057840	lsifil058302	shist058014	

auxiliary files 'hccn32' and 'trpfil', all the files have a integer counter appended to the end of the text part of the filenames. This number is the number of hours (tau) since the beginning of the AMIP integration on January 1, 1979 00 UTC. The software described in the appendix contains Fortran subroutines for converting between date-time-group and tau. For the 6 hour interval output files, the data contained is a snapshot of the model state at this forecast time. The 24 hour interval files (c3ave...) contain daily means of various model parameters averaged over every model time step for the 24 hour period ending at the tau value in the filename. Each file's contents are described in detail in the next section.

4. NOGAPS HISTORY FILE DESCRIPTIONS

The NOGAPS AMIP model history files contain a wide variety of output variables. When NOGAPS runs as a NWP model the model history output is in the form of instantaneous snapshots of the model atmosphere state every 6 hours. The NWP output is contained in the files with names beginning with the text: 'shist....', 'c3spec....', 'c3grid....', and 'lsifil....', with appropriate 'tau' values appended as shown in table 3. For the AMIP experiment the NWP history was augmented with daily means and standard deviations of several model variables and parameters. These data are contained in the 'c3ave....' files shown in Table 3. Because the fundamental I/O mechanism used for these files is Fortran direct-access, all records within a file must be the same length. The files 'shist....' and 'c3spec....' contain NOGAPS history data that are stored as *spherical harmonic coefficients and have record lengths of 25936 bytes.* The files 'c3grid....', 'lsifil....', and 'c3ave....' contain NOGAPS history data on the NOGAPS Gaussian transform grid and have record lengths of 115216 bytes. Tables 4, 5, 6, 7, and 8 give the itemized contents of 'shist....', 'c3spec....', 'c3grid....', 'lsifil....', and 'c3ave....'.

Table 4. File 'shist....' contains spherical harmonic coefficient data for two consecutive time levels at the model history output time. Record lengths are 25936 bytes.

'shist' NOGAPS Variable	record no(s).
Vorticity at current time step	1-18
Divergence at current time step	19-36
Temperature at current time step	37-54
Specific humidity at current time step	55-72
Terrain pressure at current time step	73
Vorticity at previous time step	74-91
Divergence at previous time step	92-109
Temperature at previous time step	110-127
Specific humidity at previous time step	128-145
Terrain pressure at previous time step	146

Table 5. File 'c3spec....' contains auxiliary spherical harmonic coefficient data used internally by NOGAPS. Record lengths are 25936 bytes.

'c3spec' NOGAPS Variable	record no(s).
Radiation heating rate	1-18
Laplacian of terrain geopotential	19
Terrain pressure tendency	20

Table 6. File 'c3grid....' contains 2-dimensional Gaussian grid fields of parameters describing characteristics of NOGAPS surface features and physical processes. Record length is 115216 bytes.

'c3grid' NOGAPS Variable	record no(s).
Terrain geopotential (m/sec)**2	1
Standard deviation of terrain geopotential (m/sec)**2	2
Ground temperature (K)	3
Ground wetness climotology (0-1)	4
Ground temperature climotology (K)	5
Ground wetness (0-1)	6
Surface albedo (0-1)	7
Surface rougness (m)	8
Snow depth (mm h2o)	9
Surface solar heat flux (W/m**2)	10
Surface longwave heat flux (W/m**2)	11
Surface evaporation (kg/m**2)	12
Surface moist static energy flux (W/m**2)	13
Cumulus cloud base mass flux (kg/m**2/sec)	14
Surface drag (Nts/m**2)	15
Accumulated convective precip (mm)	16
Accumulated stable precip (mm)	17
Total precipitation rate (mm/sec)	18
Top of atmosphere net solar flux (W/m**2)	19
Top of atmosphere longwave flux (W/m**2)	20
Lifting condensation level (mb above surf)	21
Top of deepest convective cloud (mb)	22
Cumulus cloud fraction (0-1)	23
Unused	24

Table 7. File 'lsifil....' contains the NOGAPS land/sea/ice table.
Record length is 115216 bytes.

'lsifil' NOGAPS Variable	record no(s).
Opensea/bareland/seaice/landice (0:1:2:3)	1

Besides the tau-tagged history files, Table 3 shows two small auxiliary files, 'hccn32' and 'trpfil', which do not have a 'tau' value appended. File 'hccn32' contains the set of model configuration parameters (e.g., distribution of vertical layers) and physical constants (e.g., acceleration of gravity) which were used for the AMIP integrations. During all model restarts this file is read to ensure that these quantities are unchanged for the entire integration period.

File 'trpfil' contains three Gaussian grid fields, defined at the beginning of the forecast, used to do the reduction to sea level necessary to output fields on constant pressure surfaces. The first field is the 'pdiff' array, which is the pressure difference between sea level and terrain level at the initial forecast time, which for the AMIP integration was 1 Jan 1979 00 UTC. 'Pdiff' ranges from zero millibars over the oceans to about 500 millibars over the Himalayas. The second field is a surface to 100 millibar layer mean atmospheric temperature. Changes in this deep layer mean during the forecast are used to adjust the 'pdiff' values to allow for the effect of warming or cooling, on the fictitious subterranean air column between sea level and terrain level. The third field is the initial 1000 mb temperature. This field, combined with the second field above, are used in short term NWP forecasts to accurately predict sea level pressure changes.

Users of the NOGAPS model history will probably have their own ideas about how to do the reduction to sea level process. However, for the purposes of looking at long term means, we suggest that the 'pdiff' array above be added to time averaged terrain pressures without worrying about the subtleties of temperature correcting used in short term NWP forecasts. Sea level pressure prediction under high terrain is certainly not a priority question in long term atmospheric simulation and should be kept as simple as possible.

5. SUMMARY

Anyone with an account on the SSC computer system can access the NOGAPS AMIP history data base. All files have read access for everyone. There is no anonymous FTP access to SSC, so unfortunately outside access is not available. If non-SSC parties are interested in the AMIP data, other arrangements may be possible, however.

Table 8. File 'c3ave....' contains Gaussian grid fields of daily means and variances of several NOGAPS predicted quantities. Record length is 115216 bytes.

'c3ave' NOGAPS Variable	record no(s).
Latent heat flux (W/m**2)	1
Sensible heat flux (W/m**2)	2
Top of atmosphere solar flux (W/m**2)	3
Top of clear atmosphere solar flux (W/m**2)	4
Surface solar flux (W/m**2)	5
Surface solar flux, clear atm (W/m**2)	6
U-comp, surface drag (Nts/m**2)	7
V-comp, surface drag (nts/m**2)	8
Top of atmosphere longwave flux (W/m**2)	9
Surface longwave flux (W/m**2)	10
Top of clear atmosphere longwave flux (W/m**2)	11
Surface longwave flux, clear atm (W/m**2)	12
Surface air temperature (K)	13
Spectral truncation surface moisture flux (W/m**2)	14
Convective precipitation (mm)	15
Stable precipitation (mm)	16
Terrain pressure (mb)	17
Variance of surface air temperature (K**2)	18
Ground temperature (K)	19
Variance of ground temperature (K**2)	20
Diabatic heating (K/day)	21-38
Diabatic moistening (kg/kg/day)	39-56
Diabatic U-comp change (m/sec/day)	57-74
Diabatic V-comp change (m/sec/day)	75-92
Precipitation heating (K/day)	93-110
Precipitation moistening (kg/kg/day)	111-128
Type 2 cloud forcing stencil	129
Cumulus friction, U-comp (m/sec/day)	130-146
Cumulus friction, V-comp (m/sec/day)	147-164
Total cloud fraction (0-1)	165
Total precipitable water (kg/m**2)	166
Cumulus ensemble cloud base mass flux (kg/m**2/day)	167-182
Gravity wave drag, U-comp (Nts/m**2)	183-200
Gravity wave drag, V-comp (Nts/m**2)	201-218
Solar heating (K/day)	219-236
Longwave heating (K/day)	237-254
Solar heating, clear atmos (K/day)	255-272
Longwave heating, clear atmos (K/day)	273-290
Stable cloud fraction (0-1)	291-308
Convective cloud fraction (0-1)	309-326
Temperature change, horizontal diffusion (K/day)	327-344
Moisture change, horizontal diffusion (kg/kg/day)	345-362
U-comp change, horizontal diffusion (m/sec/day)	363-380
V-comp change, horizontal diffusion (m/sec/day)	381-398

In the following appendix several subroutines for I/O and data management and manipulation are listed. Also listed is a program used to process the data from one of the history directories into a standard set of monthly mean quantities requested by PCMDI from each of the AMIP participants. Users should find it a useful example to modify or base their own AMIP data processing program. All source code of the listed software is in the directory 'a/rosmond/pcmdi/sources'. Users are warned that this software is written for CRAY systems and has some unique requirements unlikely to be available on other platforms. Specifically, there are several references to CRAY scientific library routines such as FFTs, and the non-standard dynamic array extension is used in several subroutines. If necessary, however, the author is willing to provide some advice about software conversion.

REFERENCES

Gates, W. L., 1992: AMIP: The Atmospheric Model Intercomparison Project. Bull. Amer. Meteor. Soc., 73. 1962-1970.

Hogan, T. F., T. E. Rosmond, and R. Gelaro, 1991: The NOGAPS Forecast Model: A Technical Description. NOARL Report 13. Naval Research Laboratory, Monterey, CA., 220 pp.

APPENDIX - SOFTWARE LISTINGS

```
      subroutine cons(rad,radsq,msort,lsort,mlsort,eps4,cim,weight,sinl
     *, ocos,poly,dpoly)
c
c subroutine to create a number of essential constant operator arrays
c for NOGAPS history file processing (T47)
c
c    output:
c
c rad: radius of earth: 6371000.0
c radsq: rad*rad
c msort: zonal wavenumber(m)= msort(spherical harmonic index(ml))
c lsort: total wavenumber(l)= lsort(spherical harmonic index(ml))
c mlsort: spherical harmonic index(ml)= mlsort(m,l)
c eps4: spherical harmonic laplacian operator
c cim: fourier coefficient zonal wavenumber operator
c weight: gaussian quadrature weights
c sinl: sine of gaussian latitudes
c ocos: 1.0/cos(latitude)**2
c poly: associated legendre polynomials
c dpoly: d(poly)/d(sinl)
c
c ********************************************************************
c
      parameter (im=144, lm=18, lpout=6)
      parameter (jm= im/2, im2= im/2,  jm2= jm/2)
      parameter (jtrun= 2*((1+(im-1)/3)/2),  mlmax= (jtrun+1)*jtrun/2)
      parameter (imlm=im*lm,  imjm=im*jm,  idim2= mlmax*2)
c
      common/ fftcom/ trigs(im,2),ifax(19)
c
      dimension msort(mlmax),lsort(mlmax),mlsort(jtrun,jtrun)
     *, eps4(mlmax),cim(mlmax),weight(jm),sinl(jm),ocos(jm)
     *, poly(mlmax,jm2),dpoly(mlmax,jm2)
c
c  generate sorting arrays
c
      call sortml(jtrun,mlmax,msort,lsort,mlsort)
c
c generate laplacian operator and fourier coefficient zonal
c wavenumber coefficient
c
      rad= 6371000.0
      radsq= rad*rad
      do 150 ml=1,mlmax
      rl= lsort(ml)
      rm= msort(ml)-1
      rlm= rl-1.0
      if(msort(ml).eq.1) rm= 0.0
      if(lsort(ml).eq.1) rlm= 0.0
      eps4(ml)= rl*rlm/radsq
      cim(ml)= rm
  150 continue
c
c  generate gaussian quadrature weights and latitudes
c
      call gausl3(jm,-1.0,1.0,weight,sinl)
c
      do 155 j=1,jm
      ocos(j)= 1.0/(1.0-sinl(j)*sinl(j))
  155 continue
c
c generate legendre functions
```

```
c
      call lgndr(jm2,jtrun,mlmax,mlsort,poly,dpoly,sinl)
c
c generate CRAY library fft coefficients
c
      call fftfax(im,ifax,trigs)
c
      return
      end
```

```
      subroutine daynum(cidtg,my,mm,md,mh,mdy,mhy)
c
c  subroutine to convert a date-time-group to equivalent
c  year/month/day/hour set
c
c    input:
c
c  cidtg:  ascii date-time-group (YYMMDDHH)
c
c    output:
c
c  my: last two digits of year
c  mm: month of year
c  md: day of month
c  mh: hour of day
c  mdy: julian date (sideral day)
c  mhy: hour of month
c
c*************************************************************
c
      character*8 cidtg
c
      dimension month(12,2)
      data month/0,31,59,90,120,151,181,212,243,273,304,334,
     $           0,31,60,91,121,152,182,213,244,274,305,335/
c
      read(cidtg,1)  my,mm,md,mh
 1    format(4i2)
      j = 1
      if(mod(my,4).eq.0) j = 2
      mdy = month(mm,j)+md
      mhy = (mdy-1)*24+mh
      return
      end
```

```
      subroutine dtgfix(cidtg,cndtg,ndif)
c
c  subroutine to compute new date-time-group (dtg) from base
c  dtg and offset period in hours
c
c    input:
c
c  cidtg: input dtg (YYMMDDHH)
c  ndif: offset time in hours
c
c    output:
c
c  cndtg: output dtg
c
c*************************************************************
      character*8 cidtg, cndtg
      dimension myc(4)
      data myc/8784,3*8760/
      call daynum(cidtg,my,mm,md,mh,mdy,mhy)
c
c  increment (decrement) hour of the year (mhy) by ndif,
c   test for change of year, adjust if necessary, reform to ndtg
c
      if(my.eq.0) my= 100
      newmhy = mhy+ndif
      if(newmhy.ge.0) go to 10
 1    my = my-1
      idx = mod(my,4)+1
      newmhy = myc(idx)+newmhy
      if(newmhy.ge.0) go to 20
      go to 1
c
 10   idd = mod(my,4)+1
      if(newmhy.lt.myc(idd)) go to 20
      newmhy = newmhy-myc(idd)
      my= my+1
      go to 10
c
   20 my= mod(my,100)
      call noday(cndtg,my,newmhy)
      return
      end
```

```
      subroutine gausl3 (n,xa,xb,wt,ab)
c
c  subroutine to compute gaussian latitudes and quadrature weights
c  for gaussian quadrature integrations
c
c ********************************************************************
c
c weights and abscissas for nth order gaussian quadrature on (xa,xb).
c input arguments
c n  -the order desired (number of gaussian latitudes)
c xa -the left endpoint of the interval of integration (-1.0 for spole)
c xb -the right endpoint of the interval of integration(1.0 for npole)
c output arguments
c ab -the n calculated abscissas
c wt -the n calculated weights
c
c ********************************************************************
c
      implicit double precision (a-h,o-z)
c
      real       ab(n)      ,wt(n)
c
c machine dependent constants---
c  tol - convergence criterion for double precision iteration
c  pi  - given to 15 significant digits
c  c1  -  1/8                    these are coefficients in mcmahon"s
c  c2  -  -31/(2*3*8**2)          expansions of the kth zero of the
c  c3  -  3779/(2*3*5*8**3)       bessel function j0(x) (cf. abramowitz,
c  c4  -  -6277237/(3*5*7*8**5)   handbook of mathematical functions).
c  u   -  (1-(2/pi)**2)/4
c
      data tol/1.d-15/,pi/3.14159265358979/,u/.148678816357662/
      data c1,c2,c3,c4/.125,-.080729166666667,.246028645833333,
     1                 -1.82443876720609 /
c
c maximum number of iterations before giving up on convergence
c
      data maxit /5/
c
c arithmetic statement function for converting integer to double
c
      dbli(i) = dble(float(i))
c
      ddif = .5d0*(dble(xb)-dble(xa))
      dsum = .5d0*(dble(xb)+dble(xa))
      if (n .gt. 1) go to 101
      ab(1) = 0.
      wt(1) = 2.*ddif
      go to 107
  101 continue
      nnp1 = n*(n+1)
      cond = 1./sqrt((.5+float(n))**2+u)
      lim = n/2
c
      do 105 k=1,lim
         b = (float(k)-.25)*pi
         bisq = 1./(b*b)
c
c rootbf approximates the kth zero of the bessel function j0(x)
c
         rootbf = b*(1.+bisq*(c1+bisq*(c2+bisq*(c3+bisq*c4))))
c
c      initial guess for kth root of legendre poly p-sub-n(x)
c
         dzero = cos(rootbf*cond)
         do 103 i=1,maxit
c
```

```
            dpm2 = 1.d0
            dpm1 = dzero
c
c       recursion relation for legendre polynomials
c
            do 102 nn=2,n
               dp = (dbli(2*nn-1)*dzero*dpm1-dbli(nn-1)*dpm2)/dbli(nn)
               dpm2 = dpm1
               dpm1 = dp
  102       continue
            dtmp = 1.d0/(1.d0-dzero*dzero)
            dppr = dbli(n)*(dpm2-dzero*dp)*dtmp
            dp2pri = (2.d0*dzero*dppr-dbli(nnp1)*dp)*dtmp
            drat = dp/dppr
c
c       cubically-convergent iterative improvement of root
c
            dzeri = dzero-drat*(1.d0+drat*dp2pri/(2.d0*dppr))
            ddum= dabs(dzeri-dzero)
         if (ddum .le. tol) go to 104
            dzero = dzeri
  103     continue
          print 504
  504     format(1x,' in gausl3, convergence failed')
  104     continue
          ddifx = ddif*dzero
          ab(k) = dsum-ddifx
          wt(k) = 2.d0*(1.d0-dzero*dzero)/(dbli(n)*dpm2)**2*ddif
          i = n-k+1
          ab(i) = dsum+ddifx
          wt(i) = wt(k)
  105 continue
c
      if (mod(n,2) .eq. 0) go to 107
      ab(lim+1) = dsum
      nm1 = n-1
      dprod = n
      do 106 k=1,nm1,2
         dprod = dbli(nm1-k)*dprod/dbli(n-k)
  106 continue
      wt(lim+1) = 2.d0/dprod**2*ddif
  107 return
      end
```

```
      subroutine lgndr(jm2,jtrun,mlmax,mlsort,poly,dpoly,sinl)
c
c  ** neprf ***, programmer tom rosmond, august 3, 1987
c
c  generate legendre polynomials and their derivatives on the
c  gaussian latitudes
c
c ref= belousov, s. l., 1962= tables of normalized associated
c        legendre polynomials. pergamon press, new york
c
c ***********************************************************
c
c    input:
c
c  jm2: number of gaussian latitudes between spole and equator
c  jtrun: zonal wavenumbers (48 for T47 model)
c  mlmax: number of degrees of freedom in spherical harmonic
c         variables = jtrun*(jtrun+1)
c  mlsort: sorting array for locating spherical harmonics as a
c          function of zonal waveno(m) and total waveno(l)
c  sinl: sine of gaussian latitudes
c
c    output:
c
c  poly: associated legendre polynomials
c  dpoly: d(poly)/d(sinl)
c
c ***************************************************************
c
      dimension poly(mlmax,jm2),dpoly(mlmax,jm2),sinl(jm2)
     *, mlsort(jtrun,jtrun)
c
      dimension pnm(49,48),dpnm(49,48)
c
c sinl is sin(latitude) = cos(colatitude)
c pnm(np,mp) is legendre polynomial p(n,m) with np=n+1, mp=m+1
c pnm(mp,np+1) is x derivative of p(n,m) with np=n+1, mp=m+1
c
      jtrunp= jtrun+1
      do 1001 j=1,jm2
      xx= sinl(j)
      sn= sqrt(1.0-xx*xx)
       sn2i = 1.0/(1.0 - xx*xx)
      rt2= sqrt(2.0)
       c1 = rt2
c
       pnm(1,1) = 1.0/rt2
      theta=-atan(xx/sqrt(1.0-xx*xx))+2.0*atan(1.0)
c
      do 20 n=1,jtrun
       np = n + 1
      fn=n
       fn2 = fn + fn
       fn2s = fn2*fn2
c eq 22
      c1= c1*sqrt(1.0-1.0/fn2s)
      c3= c1/sqrt(fn*(fn+1.0))
       ang = fn*theta
       s1 = 0.0
       s2 = 0.0
       c4 = 1.0
       c5 = fn
       a = -1.0
       b = 0.0
c
      do 27 kp=1,np,2
       k = kp - 1
```

```
      s2= s2+c5*sin(ang)*c4
      if (k.eq.n) c4 = 0.5*c4
      s1= s1+c4*cos(ang)
       a = a + 2.0
       b = b + 1.0
      fk=k
       ang = theta*(fn - fk - 2.0)
       c4 = (a*(fn - b + 1.0)/(b*(fn2 - a)))*c4
       c5 = c5 - 2.0
   27 continue
c eq 19
       pnm(np,1) = s1*c1
c eq 21
       pnm(np,2) = s2*c3
   20 continue
c
      do 4 mp=3,jtrunp
       m = mp - 1
      fm= m
       fm1 = fm - 1.0
       fm2 = fm - 2.0
       fm3 = fm - 3.0
      c6= sqrt(1.0+1.0/(fm+fm))
c eq 23
       pnm(mp,mp) = c6*sn*pnm(m,m)
      if (mp - jtrunp) 3,4,4
    3 continue
       nps = mp + 1
c
      do 41 np=nps,jtrunp
       n = np - 1
      fn= n
       fn2 = fn + fn
       c7 = (fn2 + 1.0)/(fn2 - 1.0)
       c8 = (fm1 + fn)/((fm + fn)*(fm2 + fn))
      c= sqrt((fn2+1.0)*c8*(fm3+fn)/(fn2-3.0))
      d= -sqrt(c7*c8*(fn-fm1))
      e= sqrt(c7*(fn-fm)/(fn+fm))
c eq 17
       pnm(np,mp) = c*pnm(np-2,mp-2)
     1            + xx*(d*pnm(np-1,mp-2) + e*pnm(np - 1,mp))
   41 continue
    4 continue
c
      do 50 mp=1,jtrun
      fm= mp-1.0
       fms = fm*fm
      do 50 np=mp,jtrun
      fnp= np
       fnp2 = fnp + fnp
       cf = (fnp*fnp - fms)*(fnp2 - 1.0)/(fnp2 + 1.0)
      cf= sqrt(cf)
c der
      dpnm(np,mp)   = -sn2i*(cf*pnm(np+1,mp) - fnp*xx*pnm(np,mp))
   50 continue
c
      do 71 m=1,jtrun
      do 71 l=m,jtrun
      ml= mlsort(m,l)
      poly(ml,j)= pnm(l,m)
      dpoly(ml,j)=dpnm(l,m)
   71 continue
      dpoly(1,j)= 0.0
 1001 continue
      return
      end
```

```
      subroutine nfopen(filnam,msg,iread,itau,itype,len,iun,file,istat)
c
c  subroutine to open NOGAPS history files
c
c    input:
c
c  filnam:  path/filename
c  msg: flag to control diagnostic prints
c  iread: flag set if opening with first access to read (returns
c         bad status if no prexisting file)
c  itau: forecast time in hours (if itau lt 0, tau is irelevant)
c  itype: flag for 32 bit vs 64 bit I/O (itype .ne. 0: 32 bit)
c  len: number of data elements (words) to records
c
c    output:
c
c  filnam: input filnam with itau (6 digits) appended
c  istat: status (ne 0: problems with open)
c
c  ****************************************************************
c
      character*54 file
      character*48 filnam
      character*8 status
      character*6 ctau
c
      logical lex,opn,iread
c
      lenr= 8*(2+len)
      if(itype.ne.0) lenr= 8*(2+(len+1)/2)
c
      call chlen(filnam,lenc)
      write(ctau,'(i6.6)') itau
c
      if(itau.lt.0) then
      file= filnam(1:lenc)
      else
      file= filnam(1:lenc)//ctau
      endif
c
      inquire(file=file,exist=lex,opened=opn)
      if(opn) return
c
      if(lex) then
      status='old'
      istat= 0
      else
      status='new'
c
      if(iread) then
      istat=2
      return
      endif
c
      endif
c
      open(unit=iun,file=file,access='direct',form='unformatted'
     *, recl=lenr,status=status)
c
      if(msg.lt.2) return
c
      print 100, status(1:3),filnam,iun,lenr
  100 format(1x,a3,1x,a54,' opened as unit=',i3,' : lenr=',i9,' bytes ')
      return
      end
      subroutine nfread(fnam,msg,itype,istrt,nrec,len,x,itau,cdtg,istat)
c
```

```
c  subroutine to read NOGAPS history files (but can also be used for
c  general purpose IO).  file structure is FORTRAN direct access.  as an
c  option can retrieve data written in IEEE format (e.g. SUN format) and
c  convert it to CRAY 64 bit data.
c
c  formal parameters
c
c  INPUT:
c
c  fnam: character*48 path/filename to data to be read.  path can be
c    relative or absolute but must be pre-existing.  actual filename can
c    have a indexing string appended depending of value of formal parameter
c    'itau'.  see example below.
c
c  msg: flag to control information prints about file activity
c       =0, no prints
c       =1, print info about read or write action
c       =2, print info about file opening, reading and writing action
c       = anything else, same as 2
c
c  itype:  data type
c       = 0, do not unpack data, used if file data is already CRAY format
c       = 1 (or -1), unpack from IEEE integer to CRAY integer
c       = 2 (or -2), unpack from IEEE float pt. to CRAY float pt.
c
c  istrt: beginning direct access index of records to be read from fnam
c
c  nrec: number of records (beginning at istrt) to be read from fnam
c
c  len:  number of data items read into array x. (note: since
c    file format is direct access, len must be the same for all records
c    on a given fnam)
c
c  itau: counter index.  if .GE. zero it is appended as a character*6
c    string to fnam, yielding a new filename which is the actual
c    one opened for the requested data.  see example below.
c
c  OUTPUT:
c
c  x: array that data is being read into. should have dimension at least
c    (nrec*len)
c
c  itau: counter index. integer item included with data read and returned to
c  calling program.  the returned value of itau can be compared to the input
c    value to check data integrity.
c
c  cdtg: data-time-group character*8 item of data read and returned to calling
c    program.
c
c  istat: returned status code.
c       =0, no problems
c       =1, bad read, data not returned
c       =2  file missing, no data returned
c
c  EXAMPLE:
c
c ***************************************************************************
c
!     program test
c
c     program to demonstrate use of subroutines nfread and nfwrits
c     remove !'s from column 1 to get compilible code
c
!     parameter (len= 10001)
!     dimension x(len,5)
!     dimension xfull(len,5)
!     dimension ix(len,5)
```

```
!     character*48 ifile
!     character*8 cdtg
c
c  change the following path/filename to suit your interests
c
!     ifile='/a/yourname/datapath/dummy'
!     cdtg='92030412'
!     n=5
!     itau= 10
c
c  generate some test data
c
!     do 1 j=1,n
!     do 1 i=1,len
!     x(i,j)= (i+j)
!     x(i,j)=-x(i,j)**3.888
!     xfull(i,j)= x(i,j)
!     ix(i,j)= (i+j)**2
!    1 continue
c
c  this is the largest integer preserved when packed and
c  unpacked. put it at the end of the ix array.
c
!     ix(len,5)=( 2**31-1)
c
!     print*,' *** case 1 ***'
c
c  write some packed fl. pt. data and read it back in.
c  itau is .GE. 0, so it is appended to ifile, itype is
c  .GT. 0, so nfwrit returns 64 bit data with precision
c  truncated to 32 bits.
c
!     itype=  2
!     print*, ' x before packing= ',x(len,5)
!     call nfwrit(ifile,1,itype,1,n,len,x,itau,cdtg,istat)
!     print*, ' x returned from nfwrit=', x(len,5)
!     call nfread(ifile,1,itype,1,n,len,x,itau,cdtg,istat)
!     print*, ' x returned from nfread=', x(len,5)
c
!     print*,' *** case 2 ***'
c
c  write some packed integer data and read it back in
c  the data is added to ifile starting at record no. 6.
c
!     itype= 1
!     print*, ' ix before packing=', ix(len,5)
!     call nfwrit(ifile,1,itype,6,n,len,ix,itau,cdtg,istat)
!     call nfread(ifile,1,itype,6,n,len,ix,itau,cdtg,istat)
c
!     print*, ' ix after unpacking=',ix(len,5)
c
!     print*,' *** case 3 ***'
c
c  write some packed fl. pt. data and read it back in.
c  itau is .GE. 0, so it is appended to ifile, itype is
c  .LT. 0, so nfwrit returns full 64 bit data. data is
c  still written as 32 bit IEEE, however.
c
!     itype=  -2
!     print*, ' x before packing= ',x(len,5)
!     call nfwrit(ifile,1,itype,1,n,len,x,itau,cdtg,istat)
!     print*, ' x returned from nfwrit=', x(len,5)
!     call nfread(ifile,1,itype,1,n,len,x,itau,cdtg,istat)
!     print*, ' x returned from nfread=', x(len,5)
c
!     print*,' *** case 4 ***'
c
```

```
c  write some packed integer data to a file without the appended
c  counter and read it back in
c
!     itau= -1
!     itype= 1
!     print*, ' ix before packing=', ix(len,5)
!     call nfwrit(ifile,1,itype,1,n,len,ix,itau,cdtg,istat)
!     call nfread(ifile,1,itype,1,n,len,ix,itau,cdtg,istat)
c
!     print*, ' ix after unpacking=',ix(len,5)
c
!     print*,' *** case 5 ***'
c
c  write some unpacked fl. pt. data to a file and read it back in.
c  note that a different value of positive itau generates a new
c  filename which can have the different length records the full
c  precision array requires
c
!     itau= 20
!     itype= 0
!     print*, ' xfull before write= ',xfull(len,5)
!     call nfwrit(ifile,1,itype,1,n,len,xfull,itau,cdtg,istat)
!     print*, ' xfull returned from nfwrit=', xfull(len,5)
!     call nfread(ifile,1,itype,1,n,len,xfull,itau,cdtg,istat)
!     print*, ' xfull returned from nfread=', xfull(len,5)
c
c  after running this test example look at contents of your
c  target directory to verify filenames and sizes.
c
!     stop
!     end
c
c ***END OF EXAMPLE*********************************************
c
      dimension x(len,nrec)
      dimension xpack((len+1)/2)
c
      character*8 cdtg
      character*48 fnam,blk24
      character*54 file
c
      parameter (nf=20)
      common/fncom/ cfiles(nf)
      common/fucom/ fcall
      character*48 cfiles
      logical fcall,iread
c
      data blk24/'                                                '/
      data fcall/.true./
      if(fcall) then
      fcall=.false.
      do 5 i=1,nf
      cfiles(i)=blk24
    5 continue
      endif
c
      do 10 k=1,nf
      kk= k
      if(cfiles(k).eq.blk24) cfiles(k)= fnam
      if(cfiles(k).eq.fnam) go to 20
   10 continue
   20 iun= 10+kk
c
c
      iread=.true.
      call nfopen(fnam,msg,iread,itau,itype,len,iun,file,istat)
c
```

```
      if(istat.eq.2) then
      print 500, fnam,istat
  500 format(1x,a54,' missing, istat=',i1)
      return
      endif
c
      if(itype.eq.0) then
c
c  no unpacking
c
      do 30 j=1,nrec
      irec= istrt+j-1
      read(iun,rec=irec,err=45) (x(i,j),i=1,len),itau,cdtg
   30 continue
c
      else
c
c  data is packed, convert from ieee 32 bit to cray 64 bit
c
      itp= abs(itype)
      lenx= (len+1)/2
      do 40 j=1,nrec
      irec= istrt+j-1
      read(iun,rec=irec,err=45) (xpack(i),i=1,lenx),itau,cdtg
      ierr= ieg2cray(itp,len,xpack,0,x(1,j))
   40 continue
c
      endif
c
      if(itau.ge.0) close(unit=iun)
c
      istat= 0
      if(msg.eq.0) return
c
      print 100,file,nrec,istrt,len
  100 format(1x,a54,'read: ',i3,' records starting at rec=',i4
     *,' : len= ',i8)
c
      if(itype.eq.0) then
      print 200, itau,cdtg
  200 format(' data read for tau=',i6,' : dtg= ',a8)
      else
      print 300, itype,itau,cdtg
  300 format(' type',i2,' packed data read for tau=',i6,' : dtg= ',a8)
      endif
c
      return
c
   45 print 400, file,iun,irec
  400 format(' bad read on ',a54,' : unit=',i3,' : record= ',i4)
      istat= 1
      return
      end
```

```
      subroutine nfwrit(fnam,msg,itype,istrt,nrec,len,x,itau,cdtg,istat)
c
c  subroutine to write NOGAPS history files (but can also be used for
c  general purpose IO).  file structure is FORTRAN direct access.  as an
c  option can take  CRAY 64 bit data and convert it to 32 bit IEEE format
c  (e.g. SUN format) before writing the data.  files are opened internally,
c  with up to 20 files opened simulaneously, using units 11-30.  depending
c  on actual number of files opened by nfread/nfwrit, users must ensure
c  that needed unit numbers are not used elsewhere in their program.
c
c  formal parameters
c
c  INPUT:
c
c  fnam: character*48 path/filename to data to be written.  path can
c    relative or absolute but must be pre-existing.  actual filename may
c    have an indexing string appended depending of value of formal parameter
c    'itau'.  see example below.
c
c  msg: flag to control information prints about file activity
c       =0, no prints
c       =1, print info about read or write action
c       =2, print info about file opening, reading and writing action
c       = anything else, same as 2
c
c  itype:  data type
c       = 0, do not upack data, used if file data is already CRAY format
c       = 1 (or -1), pack from CRAY integer to IEEE integer
c       = 2 (or -2), pack from CRAY float pt. to IEEE float pt.
c     if itype .GT. 0 then 64 bit data words returned in array x are truncated
c     back to 32 bit precision.
c
c  istrt: beginning direct access index of records written to fnam
c
c  nrec: number of records (beginning at istrt) written to fnam
c
c  len:   number of data items in array x for fnam records. (note: since
c    file format is direct access, len must be the same for all records
c    on a given fnam)
c
c  x: array containing data to be written. should have dimension at least
c    (nrec*len)
c
c  itau: counter index.  if .GE. to zero it is appended as a character*6
c    string to fnam, yielding a new filename which is the actual
c    one opened.  itau is also written out as part of data record.
c    see example below.
c
c  cdtg: data-time-group character*8 item of data written as part of data
c    record
c
c  OUTPUT:
c
c  istat: returned status code.
c       =0, no problems
c       =1, bad write, data not output
c
c  EXAMPLE:
c
c ***************************************************************************
c
!     program test
c
c     program to demonstrate use of subroutines nfread and nfwrits
c     remove !'s from column 1 to get compilible code
c
!     parameter (len= 10001)
```

```
!     dimension x(len,5)
!     dimension xfull(len,5)
!     dimension ix(len,5)
!     character*48 ifile
!     character*8 cdtg
c
c  change the following path/filename to suit your interests
c
!     ifile='/a/yourname/datapath/dummy'
!     cdtg='92030412'
!     n=5
!     itau= 10
c
c  generate some test data
c
!     do 1 j=1,n
!     do 1 i=1,len
!     x(i,j)= (i+j)
!     x(i,j)=-x(i,j)**3.888
!     xfull(i,j)= x(i,j)
!     ix(i,j)= (i+j)**2
!    1 continue
c
c  this is the largest integer preserved when packed and
c  unpacked. put it at the end of the ix array.
c
!     ix(len,5)=( 2**31-1)
c
!     print*,' *** case 1 ***'
c
c  write some packed fl. pt. data and read it back in.
c  itau is .GE. 0, so it is appended to ifile, itype is
c  .GT. 0, so nfwrit returns 64 bit data with precision
c  truncated to 32 bits.
c
!     itype=  2
!     print*, ' x before packing= ',x(len,5)
!     call nfwrit(ifile,1,itype,1,n,len,x,itau,cdtg,istat)
!     print*, ' x returned from nfwrit=', x(len,5)
!     call nfread(ifile,1,itype,1,n,len,x,itau,cdtg,istat)
!     print*, ' x returned from nfread=', x(len,5)
c
!     print*,' *** case 2 ***'
c
c  write some packed integer data and read it back in
c  the data is added to ifile starting at record no. 6.
c
!     itype= 1
!     print*, ' ix before packing=', ix(len,5)
!     call nfwrit(ifile,1,itype,6,n,len,ix,itau,cdtg,istat)
!     call nfread(ifile,1,itype,6,n,len,ix,itau,cdtg,istat)
c
!     print*, ' ix after unpacking=',ix(len,5)
c
!     print*,' *** case 3 ***'
c
c  write some packed fl. pt. data and read it back in.
c  itau is .GE. 0, so it is appended to ifile, itype is
c  .LT. 0, so nfwrit returns full 64 bit data. data is
c  still written as 32 bit IEEE, however.
c
!     itype=  -2
!     print*, ' x before packing= ',x(len,5)
!     call nfwrit(ifile,1,itype,1,n,len,x,itau,cdtg,istat)
!     print*, ' x returned from nfwrit=', x(len,5)
!     call nfread(ifile,1,itype,1,n,len,x,itau,cdtg,istat)
!     print*, ' x returned from nfread=', x(len,5)
```

```
c
!     print*,' *** case 4 ***'
c
c  write some packed integer data to a file without the appended
c  counter and read it back in
c
!     itau= -1
!     itype= 1
!     print*, ' ix before packing=', ix(len,5)
!     call nfwrit(ifile,1,itype,1,n,len,ix,itau,cdtg,istat)
!     call nfread(ifile,1,itype,1,n,len,ix,itau,cdtg,istat)
c
!     print*, ' ix after unpacking=',ix(len,5)
c
!     print*,' *** case 5 ***'
c
c  write some unpacked fl. pt. data to a file and read it back in.
c  note that a different value of positive itau generates a new
c  filename which can have the different length records the full
c  precision array requires
c
!     itau= 20
!     itype= 0
!     print*, ' xfull before write= ',xfull(len,5)
!     call nfwrit(ifile,1,itype,1,n,len,xfull,itau,cdtg,istat)
!     print*, ' xfull returned from nfwrit=', xfull(len,5)
!     call nfread(ifile,1,itype,1,n,len,xfull,itau,cdtg,istat)
!     print*, ' xfull returned from nfread=', xfull(len,5)
c
c  after running this test example look at contents of your
c  target directory to verify filenames and sizes.
c
!     stop
!     end
c
c ***END OF EXAMPLE*********************************************
c
      dimension x(len,nrec)
      dimension xpack((len+1)/2)
c
      character*8 cdtg
      character*48 fnam,blk24
      character*54 file
c
      parameter (nf=20)
      common/fncom/ cfiles(nf)
      common/fucom/ fcall
      character*48 cfiles
      logical fcall,iread
c
      data blk24/'                                                '/
      data fcall/.true./
      if(fcall) then
      fcall=.false.
      do 5 i=1,nf
      cfiles(i)=blk24
    5 continue
      endif
c
      do 10 k=1,nf
      kk= k
      if(cfiles(k).eq.blk24) cfiles(k)= fnam
      if(cfiles(k).eq.fnam) go to 20
   10 continue
   20 iun= 10+kk
c
      itp= abs(itype)
```

```
      iread=.false.
      call nfopen(fnam,msg,iread,itau,itp,len,iun,file,istat)
c
      if(itype.eq.0) then
c
c  write data as is, no packing
c
      do 30 j=1,nrec
      irec= istrt+j-1
      write(iun,rec=irec,err=45) (x(i,j),i=1,len),itau,cdtg
   30 continue
c
      else
c
c  pack data, cray 64 bit to ieee 32 bit
c
      lenc= (len+1)/2
      itp= abs(itype)
      do 40 j=1,nrec
      irec= istrt+j-1
      if(itype.eq.-2) call undflo(x(1,j),len)
      ierr= cray2ieg(itp,len,xpack,0,x(1,j))
      write(iun,rec=irec,err=45) (xpack(i),i=1,lenc),itau,cdtg
      if(itype.gt.1) ierr= ieg2cray(itp,len,xpack,0,x(1,j))
   40 continue
c
      endif
c
      if(itau.ge.0) close(unit= iun)
c
      istat= 0
      if(msg.eq.0) return
c
      print 100,file,nrec,istrt,len
  100 format(1x,a54,'write: ',i3,' records starting at rec=',i4
     *,' : len= ',i8)
c
      if(itype.eq.0) then
      print 200, itau,cdtg
  200 format(' data written for tau=',i6,' : dtg= ',a8)
      else
      print 300, itype,itau,cdtg
  300 format(' type',i2,' packed data written for tau=',i6,' : dtg=',a8)
      endif
c
      return
c
   45 print 400, file,iun,irec
  400 format(' bad write on ',a54,' : unit=',i3,' : record= ',i4)
      istat= 1
      return
      end
```

```
      subroutine noday(cidtg,my,mhy)
c
c  subroutine to build date-time-group from hour of year
c
c  input
c     my = last 2 digits of year
c     mhy= hour of the year
c
c  output
c
c  return ascii dtg (yymmddhh) in idtg
c
c ***********************************************************
c
      character*8 cidtg
      dimension month(12,2)
      data month/0,31,59,90,120,151,181,212,243,273,304,334,
     $           0,31,60,91,121,152,182,213,244,274,305,335/
c
      j = 1
      mdy = mhy/24+1
      if(mod(my,4).eq.0) j = 2
      kk=0
      do 2 i = 2,12
      kk=kk+1
      if(mdy.le.month(i,j)) go to 3
 2    continue
      kk=kk+1
 3    mm = kk
      mh = mod(mhy,24)
      md = mdy-month(mm,j)
      write(cidtg,1)  my,mm,md,mh
    1 format(4i2)
      do 5 i = 1,8
      if(cidtg(i:i).eq.' ') cidtg(i:i)='0'
 5    continue
c
      return
      end
```

```
      subroutine sortml(jtrun,mlmax,msort,lsort,mlsort)
c
c  subroutine to generate sorting arrays for locating elements
c  in NOGAPS spherical harmonics arrays
c
c    input:
c
c  jtrun: truncation limit: =48 for T47 model
c  mlmax: (jtrun+1)*jtrun/2
c
c    output:
c
c  msort: zonal waveno(m)= msort(ml), ml=1,mlmax
c  lsort: total waveno(l)= lsort(ml), ml=1,mlmax
c  mlsort: triangular truncation spherical harmonic
c          index(ml)= mlsort(m,l)
c
c ***************************************************
c
      dimension msort(mlmax),lsort(mlmax),mlsort(jtrun,jtrun)
c
      mlx= (jtrun/2)*((jtrun+1)/2)
      ml= 0
      do 1 k=1,jtrun-1,2
      do 1 m=1,jtrun-k
      ml= ml+1
      mlp= ml+mlx
      mlsort(m,m+k)= mlp
      mlsort(m,m+k-1)= ml
      msort(ml)= m
      lsort(ml)= m+k-1
      msort(mlp)= m
      lsort(mlp)= m+k
    1 continue
c
      ml= mlp
      do 2 m=2,jtrun,2
      ml= ml+1
      mlsort(m,jtrun)= ml
      msort(ml)= m
      lsort(ml)= jtrun
    2 continue
      return
      end
```

```
      subroutine tothrs(idtg,ichour)
c
c   subroutine converts dtg(idtg) to hour of century.
c
c  input:
c
c  idtg: date-time-group (YYMMDDHH)
c
c    output:
c
c  hours since midnight 1 Jan 1900
c
c  ************************************************************
c
      character*8 idtg,icdtg
      read(idtg,'(4i2)')iyy,imm,idd,ihh
      icdtg=idtg
      ichour=0
      iyear=0
c
c  compare for current year ,jump to final calculations.
c
  100 if(iyear .eq. iyy) go to 500
      jday=365
c
c  check to year at 00 . 1900 was not a leep year so leep test
c  isn't executed.
c  compare current year to override leap year check.
c  allows to subroutine to execute till the year 2040.
c  **** please note changing below line will allow subroutine
c       to execute longer of shorter.depending of interger given.
c       in the year 2100 deleting the following two lines will allow
c       this subroutine to run till the year 2400.
c
      if(iyy .lt. 40) goto 300
c
c  check for leap year,if yes, add 1 day to jdays(days in the year)
c
  300 if(mod(iyear,4) .eq. 0) jday=jday+1
  200 ihour=jday*24
      iyear = iyear+1
      ichour=ichour+ihour
      go to 100
c
c  calculate current years hour.
c
  500 call daynum(icdtg,iyy,imm,idd,ih,idanum,ihh)
      ichour=ichour+ihh
      return
      end
```

```
      subroutine tranrs(istrt,ll,poly,w,r,s)
c
c  subroutine to transform T47 NOGAPS gaussian grid fields
c  to spherical harmonic arrays
c
c     input:
c
c  istrt: quadrature initialization flag
c        eq 0, initialize array 's' from zero
c        ne 0, add contributions to pre-existing values in 's'
c  ll: number of levels(layers) of input/ouput arrays
c  poly: associated legendre polynomials (from subroutine lgndr)
c  w: gaussian quadrature weights (from subroutine gausl3)
c  r: input array of gaussian grid fields to transform
c
c     output:
c
c  s: output array of spherical harmonic coeff fields
c
c ***************************************************************
c
      parameter (im=144, lm=18, lpout=6)
      parameter (jm= im/2, im2= im/2,  jm2= jm/2)
      parameter (jtrun= 2*((1+(im-1)/3)/2),  mlmax= (jtrun+1)*jtrun/2)
      parameter (imlm=im*lm,  imjm=im*jm,  idim2= mlmax*2)
c
      parameter (mlx= (jtrun/2)*((jtrun+1)/2))
      parameter (jump= im+3)
c
      dimension w(jm),poly(idim2,jm2),s(mlmax,2,ll),r(im,jm,ll)
      dimension cc(jump,jm),work(imjm,2)
c
      common/ fftcom/ trigs(im,2),ifax(19)
c
      jstrt= 1
      if(istrt.eq.0) jstrt= 2
c
      do 30 k=1,ll
      do 23 j=1,jm
      do 23 i=1,im
      cc(i,j)= r(i,j,k)
   23 continue
c
      call rfftmlt(cc,work,trigs,ifax,1,jump,im,jm,-1)
c
c  if istrt .eq. zero, the quadrature integral is initialized from zero,
c  otherwise the sum is added to initial 's' array passed with call.
c
      if(istrt.eq.0) then
c
      m1= 0
      do 62 l=jtrun-1,1,-2
CDIR@    IVDEP
         do 63 m = 1, l
         mm= 2*m-1
         mp= mm+1
         ml= m+m1
         mk= ml+mlx
            s(ml,1,k) = w(1)*poly(ml,1)*(cc(mm,1)+cc(mm,jm))
            s(ml,2,k) = w(1)*poly(ml,1)*(cc(mp,1)+cc(mp,jm))
            s(mk,1,k) = w(1)*poly(mk,1)*(cc(mm,1)-cc(mm,jm))
            s(mk,2,k) = w(1)*poly(mk,1)*(cc(mp,1)-cc(mp,jm))
   63    continue
      m1= m1+l
   62 continue
c
      ml= mlx*2
```

```
CDIR@ IVDEP
      do 64 m=2,jtrun,2
      ml=ml+1
      mm= 2*m-1
      mp= mm+1
      s(ml,1,k)= w(1)*poly(ml,1)*(cc(mm,1)+cc(mm,jm))
      s(ml,2,k)= w(1)*poly(ml,1)*(cc(mp,1)+cc(mp,jm))
   64 continue
      endif
c
      do 70 j=jstrt,jm2
      jj= jm-j+1
      m1= 0
      do 72 l=jtrun-1,1,-2
CDIR@    IVDEP
         do 73 m = 1, l
         mm= 2*m-1
         mp= mm+1
         ml= m+m1
         mk= ml+mlx
            s(ml,1,k) = s(ml,1,k)+w(j)*poly(ml,j)*(cc(mm,j)+cc(mm,jj))
            s(ml,2,k) = s(ml,2,k)+w(j)*poly(ml,j)*(cc(mp,j)+cc(mp,jj))
            s(mk,1,k) = s(mk,1,k)+w(j)*poly(mk,j)*(cc(mm,j)-cc(mm,jj))
            s(mk,2,k) = s(mk,2,k)+w(j)*poly(mk,j)*(cc(mp,j)-cc(mp,jj))
   73    continue
      m1= m1+l
   72 continue
c
      ml= mlx*2
CDIR@ IVDEP
      do 65 m=2,jtrun,2
      ml=ml+1
      mm= 2*m-1
      mp= mm+1
      s(ml,1,k)= s(ml,1,k)+w(j)*poly(ml,j)*(cc(mm,j)+cc(mm,jj))
      s(ml,2,k)= s(ml,2,k)+w(j)*poly(ml,j)*(cc(mp,j)+cc(mp,jj))
   65 continue
   70 continue
   30 continue
c
      return
      end
```

```
      subroutine transr(ll,poly,s,r)
c
c  subroutine to transform a T47 NOGAPS spherical harmonic coefficient
c  arrays to equivalent gaussian grid gridpoint arrays
c
c     input:
c
c  ll: number of levels(layers) in input/output arrays
c  poly: associated legendre polynomials (from subroutine lgndr)
c  s: spherical harmonic arrays
c
c     output:
c
c  r: gaussian grid arrays
c
c ************************************************************
c
      parameter (im=144, lm=18, lpout=6)
      parameter (jm= im/2, im2= im/2,  jm2= jm/2)
      parameter (jtrun= 2*((1+(im-1)/3)/2),  mlmax= (jtrun+1)*jtrun/2)
      parameter (imlm=im*lm,  imjm=im*jm,  idim2= mlmax*2)
c
      parameter (mlx= (jtrun/2)*((jtrun+1)/2))
      parameter (jump= im+3)
c
      dimension poly(idim2,jm2),s(mlmax,2,ll),r(im,jm,ll)
      dimension cc(jump,jm),work(imjm,2)
c
      common/ fftcom/ trigs(im,2),ifax(19)
c
      do 20 k=1,ll
      do 55 m=1,jump*jm2
      cc(m,1)= 0.0
      cc(m,jm2+1)= 0.0
   55 continue
c
      do 5 j=1,jm2
      jj= jm+1-j
      ml= 2*mlx
CDIR@ IVDEP
      do 3 m=2,jtrun,2
      ml= ml+1
      mm= 2*m-1
      mp= mm+1
      cc(mm,j)= poly(ml,j)*s(ml,1,k)
      cc(mp,j)= poly(ml,j)*s(ml,2,k)
      cc(mm,jj)= cc(mm,j)
      cc(mp,jj)= cc(mp,j)
    3 continue
c
      m1= 0
      do 5 l=jtrun-1,1,-2
CDIR@ IVDEP
      do 6 m=1,l
      mm= 2*m-1
      mp= mm+1
      ml= m+m1
      mk= ml+mlx
      cc(mm,j)= cc(mm,j)+poly(ml,j)*s(ml,1,k)+poly(mk,j)*s(mk,1,k)
      cc(mm,jj)=cc(mm,jj)+poly(ml,j)*s(ml,1,k)-poly(mk,j)*s(mk,1,k)
      cc(mp,j)= cc(mp,j)+poly(ml,j)*s(ml,2,k)+poly(mk,j)*s(mk,2,k)
      cc(mp,jj)=cc(mp,jj)+poly(ml,j)*s(ml,2,k)-poly(mk,j)*s(mk,2,k)
    6 continue
      m1= m1+l
    5 continue
c
      call rfftmlt(cc,work,trigs,ifax,1,jump,im,jm,1)
```

```
c
      do 22 j=1,jm
      do 22 i=1,im
      r(i,j,k)= cc(i,j)
   22 continue
   20 continue
c
      return
      end
```

```
      subroutine tranuv(ll,radsq,ocos,eps4,cim,poly,dpoly
     *, vor,div,ut,vt)
c
c  subroutine to transform NOGAPS T47 spherical harmonic arrays of
c  vorticity and divergence into gaussian grid u and v components
c
c     input:
c
c  ll: number of levels(layers) of input and output fields
c  radsq: (radius of earth)**2 = (6371000.0**2)
c  ocos: 1.0/(cos(latitude)**2
c  eps4: spherical harmonic laplacian operator= l*(l+1)/radsq, where
c        l= total waveno (l=0,jtrun-1) (from subroutine cons)
c  cim: zonal wavenumber operator  (from subroutine cons)
c  poly: associated legendre polynomials (from subroutine lgndr)
c  dpoly: d(poly)/d(sin(latitude))  (from subroutine lgndr)
c  vor: spherical harmonic vorticity
c  div: spherical harmonic divergence
c
c    output:
c
c  ut: u-component (weighted by cos(latitude)/(earth radius))
c  vt: v-component (weighted by cos(latitude)/(earth radius))
c
c ************************************************************
c
      parameter (im=144, lm=18, lpout=6)
      parameter (jm= im/2, im2= im/2,  jm2= jm/2)
      parameter (jtrun= 2*((1+(im-1)/3)/2),  mlmax= (jtrun+1)*jtrun/2)
      parameter (imlm=im*lm,  imjm=im*jm,  idim2= mlmax*2)
c
      parameter (mlx= (jtrun/2)*((jtrun+1)/2))
      parameter (jump= im+3)
c
      dimension ocos(jm),eps4(mlmax),cim(mlmax),poly(idim2,jm2)
     *, dpoly(idim2,jm2),vor(mlmax,2,lm)
     *, div(mlmax,2,lm),ut(im,jm,lm),vt(im,jm,lm)
c
      dimension cu(jump,jm),cv(jump,jm),work(imjm,2)
     *, cfac(mlmax),dfac(mlmax)
      save cfac,dfac
c
      common/ fftcom/ trigs(im,2),ifax(19)
c
      logical first
      data first/.true./
c
      if(first) then
      first=.false.
CDIR@ IVDEP
      do 10 ml=2,mlmax
      dfac(ml)= 1.0/(radsq*eps4(ml))
      cfac(ml)= cim(ml)*dfac(ml)
   10 continue
      dfac(1)= 0.0
      cfac(1)= 0.0
      endif
c
      do 15 k=1,ll
      do 55 m=1,jump*jm2
      cu(m,1)= 0.0
      cu(m,jm2+1)= 0.0
      cv(m,1)= 0.0
      cv(m,jm2+1)= 0.0
   55 continue
c
      do 20 j=1,jm2
```

```
      rcos= 1.0/ocos(j)
      jj= jm+1-j
c
      ml= 2*mlx
CDIR@ IVDEP
      do 50 m=2,jtrun,2
      ml= ml+1
      mm= 2*m-1
      mp= mm+1
      cu(mm,j)=          +cfac(ml)*poly(ml,j)*div(ml,2,k)
     *                   +dfac(ml)*dpoly(ml,j)*rcos*vor(ml,1,k)
c
      cu(mm,jj)=           +cfac(ml)*poly(ml,j)*div(ml,2,k)
     *                   -dfac(ml)*dpoly(ml,j)*rcos*vor(ml,1,k)
c
      cu(mp,j)=          -cfac(ml)*poly(ml,j)*div(ml,1,k)
     *                   +dfac(ml)*dpoly(ml,j)*rcos*vor(ml,2,k)
c
      cu(mp,jj)=           -cfac(ml)*poly(ml,j)*div(ml,1,k)
     *                   -dfac(ml)*dpoly(ml,j)*rcos*vor(ml,2,k)
c
      cv(mm,j)=          +cfac(ml)*poly(ml,j)*vor(ml,2,k)
     *                   -dfac(ml)*dpoly(ml,j)*rcos*div(ml,1,k)
c
      cv(mm,jj)=           +cfac(ml)*poly(ml,j)*vor(ml,2,k)
     *                   +dfac(ml)*dpoly(ml,j)*rcos*div(ml,1,k)
c
      cv(mp,j)=          -cfac(ml)*poly(ml,j)*vor(ml,1,k)
     *                   -dfac(ml)*dpoly(ml,j)*rcos*div(ml,2,k)
c
      cv(mp,jj)=           -cfac(ml)*poly(ml,j)*vor(ml,1,k)
     *                   +dfac(ml)*dpoly(ml,j)*rcos*div(ml,2,k)
c
   50 continue
c
      m1= 0
      do 30 l=jtrun-1,1,-2
CDIR@ IVDEP
      do 40 m=1,l
      mm= 2*m-1
      mp= mm+1
      ml= m+m1
      mk= ml+mlx
c
      cu(mm,j)= cu(mm,j)+cfac(ml)*poly(ml,j)*div(ml,2,k)
     *                   +cfac(mk)*poly(mk,j)*div(mk,2,k)
     *                   +dfac(ml)*dpoly(ml,j)*rcos*vor(ml,1,k)
     *                   +dfac(mk)*dpoly(mk,j)*rcos*vor(mk,1,k)
c
      cu(mm,jj)= cu(mm,jj)+cfac(ml)*poly(ml,j)*div(ml,2,k)
     *                   -cfac(mk)*poly(mk,j)*div(mk,2,k)
     *                   -dfac(ml)*dpoly(ml,j)*rcos*vor(ml,1,k)
     *                   +dfac(mk)*dpoly(mk,j)*rcos*vor(mk,1,k)
c
      cu(mp,j)= cu(mp,j)-cfac(ml)*poly(ml,j)*div(ml,1,k)
     *                   -cfac(mk)*poly(mk,j)*div(mk,1,k)
     *                   +dfac(ml)*dpoly(ml,j)*rcos*vor(ml,2,k)
     *                   +dfac(mk)*dpoly(mk,j)*rcos*vor(mk,2,k)
c
      cu(mp,jj)= cu(mp,jj)-cfac(ml)*poly(ml,j)*div(ml,1,k)
     *                   +cfac(mk)*poly(mk,j)*div(mk,1,k)
     *                   -dfac(ml)*dpoly(ml,j)*rcos*vor(ml,2,k)
     *                   +dfac(mk)*dpoly(mk,j)*rcos*vor(mk,2,k)
c
      cv(mm,j)= cv(mm,j)+cfac(ml)*poly(ml,j)*vor(ml,2,k)
     *                   +cfac(mk)*poly(mk,j)*vor(mk,2,k)
     *                   -dfac(ml)*dpoly(ml,j)*rcos*div(ml,1,k)
```

```
     *                  -dfac(mk)*dpoly(mk,j)*rcos*div(mk,1,k)
c
      cv(mm,jj)= cv(mm,jj)+cfac(ml)*poly(ml,j)*vor(ml,2,k)
     *                  -cfac(mk)*poly(mk,j)*vor(mk,2,k)
     *                  +dfac(ml)*dpoly(ml,j)*rcos*div(ml,1,k)
     *                  -dfac(mk)*dpoly(mk,j)*rcos*div(mk,1,k)
c
      cv(mp,j)= cv(mp,j)-cfac(ml)*poly(ml,j)*vor(ml,1,k)
     *                  -cfac(mk)*poly(mk,j)*vor(mk,1,k)
     *                  -dfac(ml)*dpoly(ml,j)*rcos*div(ml,2,k)
     *                  -dfac(mk)*dpoly(mk,j)*rcos*div(mk,2,k)
c
      cv(mp,jj)= cv(mp,jj)-cfac(ml)*poly(ml,j)*vor(ml,1,k)
     *                  +cfac(mk)*poly(mk,j)*vor(mk,1,k)
     *                  +dfac(ml)*dpoly(ml,j)*rcos*div(ml,2,k)
     *                  -dfac(mk)*dpoly(mk,j)*rcos*div(mk,2,k)
c
   40 continue
c
      ml= ml+l
   30 continue
   20 continue
c
      call rfftmlt(cu,work,trigs,ifax,1,jump,im,jm,1)
      call rfftmlt(cv,work,trigs,ifax,1,jump,im,jm,1)
c
      do 22 j=1,jm
CDIR@ IVDEP
      do 22 i=1,im
      ut(i,j,k)= cu(i,j)
      vt(i,j,k)= cv(i,j)
   22 continue
c
   15 continue
      return
      end
```

```
      subroutine undflo(x,m)
c
c  subroutine to correct for illegal floating point values
c  generated when CRAY software converted underflowed IEEE
c  32 bit values back to 64 bits
c
c    input
c
c  x: input 64 bit array which may contain illegal fp values
c  m: number of elements in x
c
c    output
c
c  x: corrected values of x
c
c ********************************************************
c
      dimension x(m)
c
      do 10 i=1,m
      if(abs(x(i)).lt.1.0e-40) x(i)= 0.0
   10 continue
      return
      end
```

```
      program makeave
c
      parameter (im=144, jm=im/2, lm= 18, lpout=6)
      parameter (jtrun= 2*((1+(im-1)/3)/2), mlmax= (jtrun+1)*jtrun/2)
      parameter (idim2= 2*mlmax)
      parameter (nspec=4*lm+1)
      parameter (numave= 40+21*lpout+2+14*lpout+7)
      parameter (imjm=im*jm)
c
c
      dimension polall(mlmax,2,jm/2)
      dimension poly(idim2,jm/2),dpoly(idim2,jm/2),weight(jm)
      dimension eps4(mlmax),cim(mlmax),sinl(jm),cosl(jm),msort(mlmax)
     *, lsort(mlmax),mlsort(jtrun,jtrun),ocos(jm),onocos(im,jm)
c
      common/ fftcom/ trigs(im,2),ifax(19)
c
      equivalence (polall(1,1,1),poly)
      equivalence (polall(1,2,1),dpoly)
c
      dimension x(im*jm),pt(im,jm),avsum(im*jm),xl(im*jm,36)
     *, xl1(idim2,nspec),xl2(idim2,nspec),xl3(idim2,nspec)
     *, xl4(idim2,nspec),slpx(imjm),crfiii(imjm),dswiii(imjm)
     *, olriii(imjm),dlpl(imjm),dtpl(imjm)
c
     *, phi(imjm,lm),tt(imjm,lm+1),rvor(imjm,lm),rdiv(imjm,lm)
     *, rstrm(imjm,lm),vpot(imjm,lm)
     *, sht(imjm,lm),ut(imjm,lm),vt(imjm,lm),sgeo(imjm)
     *, plog(imjm,lm+1),pll(imjm,lm),pk(imjm,lm),pk2(imjm,lm)
c
      common/pvar/ phips(imjm,lpout),temps(imjm,lpout)
     *, sphum(imjm,lpout),psi(imjm,lpout),chi(imjm,lpout)
     *, upres(imjm,lpout),vpres(imjm,lpout),slp(imjm)
c
     *, phipsv(imjm,lpout),tempsv(imjm,lpout)
     *, sphumv(imjm,lpout),psiv(imjm,lpout),chiv(imjm,lpout)
     *, upresv(imjm,lpout),vpresv(imjm,lpout),slpv(imjm)
c
      common/spec/vornow(mlmax,2,lm),divnow(mlmax,2,lm)
     *, temnow(mlmax,2,lm),shnow(mlmax,2,lm),plnow(mlmax,2)
c
      equivalence (xl1,vornow)
c
      common/c3aves/ ave2d(imjm,36),ave3d(imjm,lpout*21)
c
      common/trp/ pdiff(imjm),tsave(imjm),tmpp1(imjm)
c
      common/cross/ tempcs(imjm,lm),shtcs(imjm,lm),rhcs(imjm,lm)
     *, cldscs(imjm,lm),uucs(imjm,lm),vvcs(imjm,lm),strmcs(imjm,lm)
     *, cstt(jm,lm),cssht(jm,lm),csrh(jm,lm)
     *, csclds(jm,lm),csuu(jm,lm),csvv(jm,lm),csstrm(jm,lm)
     *, pcs(lm),ptzm(jm)
c
      dimension glob(288*145),xin(288*145),yin(288*145)
     *, alat(jm+2)
      dimension vwk(288*145*10)
      dimension gflip(144*73)
c
      parameter (lccn= 350+lm*(12+5*lm))
c
c  ********************************************************************
c
      dimension ccn(lccn)
      equivalence (c,ccn)
      character*8 idtg
c
      common/cnstnt/ c(300),itau,ithist,idtg(3),sarray(lm,lm)
```

```
     *, carray(lm,lm),evecin(lm,lm),evectr(lm,lm),tmcor(lm,lm)
     *, eigval(lm),spalm(lm),pmcor(lm),pbcor(lm),tmean(lm)
     *, asig(lm+1),bsig(lm+1),dasig(lm),dbsig(lm)
     *, sige(lm+1),dsig(lm),pkfac(lm)
     *, ptmean,ccpad(100)
c
c  ********************************************************************
c
      equivalence
     1  (c(1),jmm1),        (c(3),jmm2),        (c(4),lncrec)
     2, (c(5),iday),        (c(6),tofday)
     3, (c(7),itauo),       (c(8),julian),      (c(9),jskip)
     4, (c(10),jsplit),     (c(11),taui),       (c(12),dtrad)
     5, (c(13),taue),       (c(14),tice),       (c(15),forwrd)
     6, (c(16),cosd),       (c(17),ddraft),     (c(18),jtrunp)
     7, (c(19),eccld),      (c(20),rbear),      (c(21),prntij)
     8, (c(22),prevap),     (c(23),radsq)
     9, (c(25),omega),      (c(26),tauave),     (c(27),pi)
     1, (c(28),pi2),        (c(29),rgas),       (c(30),capa)
     2, (c(31),cp),         (c(32),hltm),       (c(33),fim)
     3, (c(41),fluxcl),     (c(42),isunfx),     (c(43),ptop)
     4, (c(44),tauh),       (c(45),krnit),      (c(46),ls)
     5, (c(47),mombdg),     (c(48),evaprh),     (c(49),vis)
     6, (c(50),p00),        (c(51),dta),        (c(52),ops)
     7, (c(53),hybrd),      (c(54),ksgeo),      (c(55),pok)
     8, (c(56),dgtime),     (c(57),ecadv)
c
        equivalence
     1  (c(68),taud),       (c(69),dt)
     2, (c(70),pcon),       (c(71),pconkt),     (c(72),ls1)
     3, (c(73),skew),       (c(74),grav2),      (c(75),delmf)
     4, (c(76),ithfg),      (c(77),itaufg),     (c(78),lsx)
     5, (c(79),tseres),     (c(80),prftop),     (c(81),dtgpcm)
     6, (c(93),daypyr),     (c(94),day),        (c(95),rotper)
     7, (c(96),iqnx),       (c(97),perhl),      (c(98),decmax)
     8, (c(99),eccn),       (c(100),rad),       (c(101),grav)
     9, (c(102),critl),     (c(103),p608),      (c(104),lmid)
     1, (c(105),pcp),       (c(106),frqdib),    (c(107),cdtgoi)
     2, (c(109),ipfcst),    (c(110),ipsigp),    (c(111),ipinit)
     3, (c(112),ipoutp),    (c(113),tau),       (c(114),drgdry)
     4, (c(115),icnmax),    (c(116),frqlim),    (c(117),beta)
     5, (c(118),nmodev),    (c(119),dinit),     (c(120),taudb)
     6, (c(121),uvmax),     (c(123),facrad),    (c(134),gamfac)
     8, (c(135),prnt),      (c(136),capap1),    (c(137),cuprh)
c
        equivalence
     1  (c(138),novar),     (c(139),cdet),      (c(140),dcapa)
     2, (c(141),nozone),    (c(142),sstclm),    (c(143),updayc)
     3, (c(144),incrup),    (c(145),incwnd),    (c(146),pcmdi)
     4, (c(151),nsdedy),    (c(152),kmonth),    (c(153),mnthdy)
     5,                     (c(157),rsdist),    (c(158),sind)
c
c  ********************************************************************
c
c
c  ********************************************************************
c
      common/numrec/ nfws,nfss,ndsw,ncst,nss,ncss,nudg,nvdg,nulw,nrs
     *, nclt,ncls,ntsf,nqfl,ncum,nlsp,nps,nrma,ntgf,nrmg,ntcc,nth2o
     *, nevp,nctt,ncts,ncsa,ncla,ncta,npcp,nvts,nvtg,npme
     *, nsno,nwet,nwev,nalb,nblk(4)
c
     *, ntdiab,nqdiab,nudiab,nvdiab,ntcum,nqcum,nucum,nvcum
     *, ncumf,nugwd,nvgwd,ntradh,ntlwr,ntradc,ntlwrc
     *, navcld,navcvc,ndthd,ndqhd,nduhd,ndvhd
c
     *, nslp,nslpv,ntmp,ntmpv,nhgt,nhgtv,nsht,nshtv,nuwn,nuwnv
```

```
     *, nvwn,nvwnv,npsi,npsiv,nchi,nchiv
c
     *, ntmpcs,nshtcs,nrhcs,ncldcs,nuucs,nvvcs,nstmcs
c
      dimension nrecs(numave)
      equivalence (nrecs,nfws)
c
      common/reclab/ lfws,lfss,ldsw,lcst,lss,lcss,ludg,lvdg,lulw,lrs
     *, lclt,lcls,ltsf,lqfl,lcum,llsp,lps,lrma,ltgf,lrmg,ltcc,lth2o
     *, levp,lctt,lcts,lcsa,lcla,lcta,lpcp,lvts,lvtg,lpme
     *, lsno,lwet,lwev,lalb,lblk(4)
c
     *, ltdiab(lpout),lqdiab(lpout),ludiab(lpout),lvdiab(lpout)
     *, ltcum(lpout),lqcum(lpout),lucum(lpout),lvcum(lpout)
     *, lcumf(lpout),lugwd(lpout), lvgwd(lpout),ltswh(lpout)
     *, ltlwr(lpout),ltswc(lpout),ltlwrc(lpout)
     *, lavcld(lpout),lavcvc(lpout),ldthd(lpout),ldqhd(lpout)
     *, lduhd(lpout),ldvhd(lpout)
c
     *, lslp,lslpv,ltmp(lpout),ltmpv(lpout),lhgt(lpout),lhgtv(lpout)
     *, lsht(lpout),lshtv(lpout),luwn(lpout),luwnv(lpout)
     *, lvwn(lpout),lvwnv(lpout),lpsi(lpout),lpsiv(lpout)
     *, lchi(lpout),lchiv(lpout)
c
     *, ltmpcs,lshtcs,lrhcs,lcldcs,luucs,lvvcs,lstmcs
c
      character*8 lfws,lfss,ldsw,lcst,lss,lcss,ludg,lvdg,lulw,lrs
     *, lclt,lcls,ltsf,lqfl,lcum,llsp,lps,lrma,ltgf,lrmg,ltcc,lth2o
     *, levp,lctt,lcts,lcsa,lcla,lcta,lpcp,lvts,lvtg,lpme,lsno,lwet
     *, lwev,lalb,lblk
c
     *, ltdiab,lqdiab,ludiab,lvdiab,ltcum,lqcum,lucum,lvcum
     *, lcumf,lugwd,lvgwd,ltswh,ltlwr,ltswc,ltlwrc
     *, lavcld,lavcvc,ldthd,ldqhd,lduhd,ldvhd
c
     *, lslp,lslpv,ltmp,ltmpv,lhgt,lhgtv,lsht,lshtv,luwn,luwnv
     *, lvwn,lvwnv,lpsi,lpsiv,lchi,lchiv
c
     *, ltmpcs,lshtcs,lrhcs,lcldcs,luucs,lvvcs,lstmcs
c
      character*8 clrecs(numave),clrecl(21*lpout),clreca(2+14*lpout)
     *, clcros(7)
c
      equivalence (clrecs,lfws)
      equivalence (clrecl,ltdiab)
      equivalence (clreca,lslp)
      equivalence (clcros,ltmpcs)
c
      dimension pout(lpout),psout(lpout),pcsl(lm)
c
      logical superg,crf,fpass
c
      character*64 capsun
      character*36 maplst
      character*8 ident(24)
      character*8 idtgs,idtge,idtgx,idx
      character*8 lrec
      character*4 iplev
      character*48 hccn,c3ave,filarg,filsml,trpfil,c3grid,shist
      character*48 c3path,avpath
      character*24 llab(numave)
      character*2 month,newmon
      character*6 c3av,shst
      character*7 hcc,c3g,trpf
      character*64 cap(7)
c
      dimension plotc(5,numave)
```

```
c
      data zero /0.0/
      data pout/10.0,100.0,200.0,500.0,850.0,1000.0/
      data pcs/10.0,20.0,30.0,50.0,70.0,100.0,150.0,200.0,250.0
     *, 300.0,400.0,500.0,600.0,700.0,800.0,900.0,950.0,1000.0/
c
      open(2,file='/u/b/rosmond/timave/labparm',form='formatted')
c
      read(2,800) (clrecs(k),llab(k),(plotc(ki,k),ki=1,5)
     *,  k=1,numave)
  800 format(6x,a8,1x,a24,2x,4f8.3,f4.1)
c
      print 850,(clrecs(k),llab(k)
     *,(plotc(ki,k),ki=1,5),k=1,numave)
  850 format(6x,a8,1x,a24,2x,4f14.5,f4.1)
c
      isize= -1
c
      do 2 k=1,numave
      nrecs(k)= k
    2 continue
c
      do 3 k=1,lpout
      psout(k)= log(pout(k))
    3 continue
c
      do 4 k=1,lm
      pcsl(k)= log(pcs(k))
    4 continue
c
c  compute interpolation coeffs and tranpose array y
c
      pi= 4.0*atan(1.0)
      pio2= pi*0.5
      rad= 6375000.
      radsq= rad*rad
      fpass=.true.
c
      call fftfax(im,ifax,trigs)
c
      call gausl3(jm,-1.0,1.0,weight,sinl)
      do 5 j=2,jm+1
      alat(j)= asin(sinl(j-1))
    5 continue
      tem= pio2+1.0e-6
      alat(1)= -tem
      alat(jm+2)= tem
      call sortml(jtrun,mlmax,msort,lsort,mlsort)
c
      do 152 ml=1,mlmax
      rl= lsort(ml)
      rm= msort(ml)-1
      rlm= rl-1.0
      if(msort(ml).eq.1) rm= 0.0
      if(lsort(ml).eq.1) rlm= 0.0
      eps4(ml)= rl*rlm/radsq
      cim(ml)= rm
  152 continue
CDIR$ IVDEP
      do 155 j=1,jm/2
      sinl(jm+1-j)= -sinl(j)
      ocos(j)= 1.0/(1.0-sinl(j)*sinl(j))
      ocos(jm+1-j)= ocos(j)
      cosl(j)= 1.0/sqrt(ocos(j))
      cosl(jm+1-j)= cosl(j)
      do 155 i=1,im
      onocos(i,j)= ocos(j)
```

```
      onocos(i,jm-j+1)= ocos(j)
  155 continue
c
      call lgndr(jm/2,jtrun,idim2,mlsort,poly,dpoly,sinl)
c
c
      do 12 k=1,24
      ident(k)='        '
   12 continue
c
      open(1,file='pcards',form='formatted')
c
      read(1,80) filarg
      read(1,80) filsml
      read(1,80) avpath
      read(1,80) c3path
   80 format(a48)
c
      call chlen(avpath,lpath)
      call chlen(c3path,l3path)
      c3av='/c3ave'
      hcc= '/hccn32'
      trpf='/trpfil'
      c3g= '/c3grid'
      shst='/shist'
c
c     call cfopen(filarg,41760,istat)
c     call cfopen(filsml,10512,istat)
c
      read(1,100) capsun
  100 format(a64)
      print*,' capsun ',capsun
c
  120 read(1,140,end=160) maplst
  140 format(a36)
      print*,' maplst ',maplst
      iszold= isize
      if(maplst(1:8).eq.'nomodata') go to 160
      read(maplst,150) lrec,isize,idtgs,idtge
  150 format(a8,1x,i1,1x,a8,1x,a8)
      superg= isize.gt.0
      idtg(1)='79010100'
c
c  find itaus and itaue corresponding to beginning and ending
c  dtg of averaging period
c
      print*, ' idtg ',idtg(1)
      print*, ' idtgs ',idtgs
      print*, ' idtge',idtge
      call tothrs(idtg(1),itaux)
      call tothrs(idtgs,itaus)
      call tothrs(idtge,itaue)
      print*,' taux ',itaux
      print*,' taus ',itaus
      print*,' taue ',itaue
      itaus= itaus-itaux+24
      itaue= itaue-itaux+24
      print*,' taus ',itaus
      print*,' taue ',itaue
c
      month=idtgs(3:4)
      c3ave= c3path(1:l3path)//month//c3av
      hccn= avpath(1:lpath)//month//hcc
      trpfil= avpath(1:lpath)//month//trpf
      c3grid= avpath(1:lpath)//month//c3g
      shist= avpath(1:lpath)//month//shst
c
```

```
      c3ave= c3path(1:l3path)//c3av
      hccn= avpath(1:lpath)//hcc
      trpfil= avpath(1:lpath)//trpf
      c3grid= avpath(1:lpath)//
      shist= avpath(1:lpath)//shst
c
      idtgx= idtgs
c
      itx= -1
      call nfread(hccn,1,2,1,1,lccn,c,itx,idx,istat)
c
      if(fpass) then
      fpass=.false.
c
      call nfread(trpfil,1,2,1,3,imjm,pdiff,-1,idx,istat)
      call nfread(c3grid,1,2,1,1,imjm,sgeo,itaus,idx,istat)
      call qpnh(pdiff,'pdiffpdiff',1,1,144,72)
c
      call zilch(phips,imjm*(lpout*7+1)*2)
      call zilch(ave2d,imjm*36)
      call zilch(ave3d,imjm*lpout*21)
      call zilch(tempcs,imjm*lm*7)
      do 351 i=1,imjm
      dswiii(i)= 0.0001
      olriii(i)= 0.0001
  351 continue
c
      ipt= 0
      numav= 0
      do 165 itau= itaus,itaue,24
      itau1= itau-18
      itau2= itau-12
      itau3= itau-6
      itau4= itau
c
c  spectral  fields
c
      call nfread(shist,ipt,2,1,nspec,idim2,xl1,itau1,idx,istat)
      call nfread(shist,ipt,2,1,nspec,idim2,xl2,itau2,idx,istat)
      call nfread(shist,ipt,2,1,nspec,idim2,xl3,itau3,idx,istat)
      call nfread(shist,ipt,2,1,nspec,idim2,xl4,itau4,idx,istat)
c
      do 110 i=1,idim2*nspec
      vornow(i,1,1)= 0.25*(xl1(i,1)+xl2(i,1)+xl3(i,1)+xl4(i,1))
  110 continue
c
      call transr(lm,poly,vornow,rvor)
      call transr(lm,poly,divnow,rdiv)
      call tranuv(lm,radsq,ocos,eps4,cim,poly,dpoly
     *, vornow,divnow,ut,vt)
c
      do 115 k=1,lm
      vornow(1,1,k)= 0.0
      vornow(1,2,k)= 0.0
      divnow(1,1,k)= 0.0
      divnow(1,2,k)= 0.0
      do 115 ml=2,mlmax
      vornow(ml,1,k)=-1.0e-6*vornow(ml,1,k)/eps4(ml)
      divnow(ml,1,k)=-1.0e-6*divnow(ml,1,k)/eps4(ml)
      vornow(ml,2,k)=-1.0e-6*vornow(ml,2,k)/eps4(ml)
      divnow(ml,2,k)=-1.0e-6*divnow(ml,2,k)/eps4(ml)
  115 continue
      call transr(lm,poly,temnow,tt)
      call transr(lm,poly,shnow,sht)
      call transr(1,poly,plnow,pt)
      call transr(lm,poly,vornow,rstrm)
      call transr(lm,poly,divnow,vpot)
```

```
c
      call trngra(1,cim,poly,dpoly,plnow,dlpl,dtpl)
c
      call p2kapa(imjm,lm,asig,bsig,pt,pk,plog,pk2,pll)
c
c  hydrostatic equation
c
      cp= 1005.45
      pp= log(1000.0)
      do 402 i=1,imjm
      phi(i,lm) = sgeo(i)+cp*tt(i,lm)*(pk2(i,lm)-pk(i,lm))
      sht(i,lm)= sht(i,lm)/(dasig(lm)+dbsig(lm)*pt(i,1))
  402 continue
      do 405 l=1,lm-1
      do 405 i=1,imjm
      sht(i,l)= sht(i,l)/(dasig(l)+dbsig(l)*pt(i,1))
      phi(i,l)= tt(i,l+1)*(pk(i,l+1)-pk2(i,l))+tt(i,l)
     * *(pk2(i,l)-pk(i,l))
  405 continue
      do 440 lrev = 1,lm-1
      l = lm - lrev
      do 440 i=1,imjm
  440  phi(i,l)=phi(i,l+1)+phi(i,l)*cp
c
      call womega(im,jm,lm,onocos,bsig,dasig,dbsig,pt
     *, dtpl,dlpl,rdiv,ut,vt,xl)
c
c  compute specific humidity and temperature
c
      do 330 k=1,lm
      do 326 i=1,imjm
      sht(i,k)= max(sht(i,k),1.0e-6)
      tt(i,k)= tt(i,k)*pk(i,k)/(1.0+0.608*sht(i,k))
  326 continue
c
      call qsatvs(imjm,tt(1,k),pll(1,k),x)
c
      do 330 i=1,imjm
      rhcs(i,k)= rhcs(i,k)+min(1.0,sht(i,k)/x(i))
      tempcs(i,k)= tempcs(i,k)+tt(i,k)
      shtcs(i,k)= shtcs(i,k)+sht(i,k)
      uucs(i,k)= uucs(i,k)+ut(i,k)
      vvcs(i,k)= vvcs(i,k)+vt(i,k)
      strmcs(i,k)= strmcs(i,k)+xl(i,k)
      sht(i,k)= log(sht(i,k))
  330 continue
c
      kmid= 4
      call svar(imjm,lm,kmid,pt,sgeo,pdiff,tsave,phi,tt,tmpp1,slpx)
      call geotrp(im,jm,lm,lpout,psout,tt,phi,sgeo,slpx
     *, plog,xl,xl1)
c
      call avevar(imjm,lpout,xl,temps,tempsv)
      call avevar(imjm,lpout,xl1,phips,phipsv)
      call avevar(imjm,1,slpx,slp,slpv)
c
      call s2ptrp(im,jm,lm,lpout,ut,ut(1,lm),plog,xl,psout,0.5)
      call avevar(imjm,lpout,xl,upres,upresv)
      call s2ptrp(im,jm,lm,lpout,vt,vt(1,lm),plog,xl,psout,0.5)
      call avevar(imjm,lpout,xl,vpres,vpresv)
      call s2ptrp(im,jm,lm,lpout,sht,sht(1,lm),plog,xl,psout,0.0)
      do 333 k=1,lpout
      do 333 i=1,imjm
      xl(i,k)= exp(xl(i,k))*1000.0
  333 continue
      call avevar(imjm,lpout,xl,sphum,sphumv)
      call s2ptrp(im,jm,lm,lpout,rstrm,rstrm(1,lm),plog,xl,psout,0.5)
```

```
      call avevar(imjm,lpout,xl,psi,psiv)
      call s2ptrp(im,jm,lm,lpout,vpot,vpot(1,lm),plog,xl,psout,0.5)
      call avevar(imjm,lpout,xl,chi,chiv)
c
c  read ground wetness and snow
c
      call nfread(c3grid,ipt,2,6,1,imjm,xl(1,1),itau1,idx,istat)
      call nfread(c3grid,ipt,2,6,1,imjm,xl(1,2),itau2,idx,istat)
      call nfread(c3grid,ipt,2,6,1,imjm,xl(1,3),itau3,idx,istat)
      call nfread(c3grid,ipt,2,6,1,imjm,xl(1,4),itau4,idx,istat)
c
      call nfread(c3grid,ipt,2,9,1,imjm,xl(1,5),itau1,idx,istat)
      call nfread(c3grid,ipt,2,9,1,imjm,xl(1,6),itau2,idx,istat)
      call nfread(c3grid,ipt,2,9,1,imjm,xl(1,7),itau3,idx,istat)
      call nfread(c3grid,ipt,2,9,1,imjm,xl(1,8),itau4,idx,istat)
c
      do 337 i=1,imjm
      xl(i,nwet)=0.25*(xl(i,1)+xl(i,2)+xl(i,3)+xl(i,4))
      ave2d(i,nwet)=ave2d(i,nwet)+xl(i,nwet)
      ave2d(i,nsno)=ave2d(i,nsno)+0.25*(xl(i,5)+xl(i,6)+xl(i,7)+xl(i,8))
  337 continue
c
c  read 2-d daily mean history
c
      call nfread(c3ave,1,2,1,20,imjm,xl,itau,idx,istat)
c
      call solar(xl(1,nalb),idtgx,sinl,cosl)
c
      do 340 i=1,imjm
      ave2d(i,nfws)= ave2d(i,nfws)+xl(i,nfws)
      ave2d(i,nfss)= ave2d(i,nfss)+xl(i,nfss)
      ave2d(i,ndsw)= ave2d(i,ndsw)+xl(i,ndsw)
      ave2d(i,nss)= ave2d(i,nss)+xl(i,nss)
      ave2d(i,nudg)= ave2d(i,nudg)+xl(i,nudg)
      ave2d(i,nvdg)= ave2d(i,nvdg)+xl(i,nvdg)
      ave2d(i,nulw)= ave2d(i,nulw)+xl(i,nulw)
      ave2d(i,nrs)= ave2d(i,nrs)+xl(i,nrs)
      ave2d(i,ntsf)= ave2d(i,ntsf)+xl(i,ntsf)
      ave2d(i,nqfl)= ave2d(i,nqfl)+xl(i,nqfl)
      ave2d(i,ncum)= ave2d(i,ncum)+xl(i,ncum)
      ave2d(i,nlsp)= ave2d(i,nlsp)+xl(i,nlsp)
      ave2d(i,nps)= ave2d(i,nps)+xl(i,nps)
      ave2d(i,nrma)= ave2d(i,nrma)+xl(i,nrma)
      ave2d(i,ntgf)= ave2d(i,ntgf)+xl(i,ntgf)
      ave2d(i,nrmg)= ave2d(i,nrmg)+xl(i,nrmg)
      ave2d(i,nalb)= ave2d(i,nalb)+xl(i,nalb)
  340 continue
c
c  compute mean square ground and surface air temps for later
c  variance determination; also ground wetness
c
      do 345 i=1,imjm
      ave2d(i,nvts)= ave2d(i,nvts)+(xl(i,ntsf)-273.15)**2
      ave2d(i,nvtg)= ave2d(i,nvtg)+(xl(i,ntgf)-273.15)**2
      ave2d(i,nwev)= ave2d(i,nwev)+xl(i,nwet)**2
  345 continue
c
c  read 3-d daily mean history
c
      do 350 k=1,21
      kk= 21+(k-1)*lm
      call nfread(c3ave,ipt,2,kk,lm,imjm,tt,itau,idx,istat)
c
      if(k.eq.9) then
      do 352 i=1,imjm
      ave2d(i,ntcc)= ave2d(i,ntcc)+tt(i,1)
      ave2d(i,nth2o)=ave2d(i,nth2o)+tt(i,2)
```

```
  352 continue
      call zilch(tt,imjm*2)
      endif
c
      if(k.eq.7) then
c
c compute cloud forcing using method 3
c
      do 353 i=1,imjm
      crfiii(i)= tt(i,1)
      if(crfiii(i).gt.1.9) then
      ave2d(i,ncst)= ave2d(i,ncst)+xl(i,ndsw)-xl(i,ncst)
      ave2d(i,ncss)= ave2d(i,ncss)+xl(i,nss)-xl(i,ncss)
      dswiii(i)= dswiii(i)+1.0
      endif
c
c  longwave
c
      if(crfiii(i).gt.0.999) then
      ave2d(i,nclt)= ave2d(i,nclt)+xl(i,nclt)-xl(i,nulw)
      ave2d(i,ncls)= ave2d(i,ncls)+xl(i,ncls)-xl(i,nrs)
      olriii(i)= olriii(i)+1.0
      endif
      tt(i,1)= 0.0
  353 continue
      endif
c
      if(k.eq.16) then
      do 357 l= 1,lm
      do 357 i=1,imjm
      xl(i,l)= tt(i,l)
  357 continue
c
      else if(k.eq.17) then
c
      do 359 l=1,lm
      do 359 i=1,imjm
      cldscs(i,l)= cldscs(i,l)+max(xl(i,l),tt(i,l))
  359 continue
c
c     else if(k.eq.15) then
      endif
c
      call fixdat(k,tt,tt,imjm,lm)
c
c
      call s2ptrp(im,jm,lm,lpout,tt,tt(1,lm),plog,rvor,psout,0.5)
c
      do 355 j=1,lpout
      jj= j+(k-1)*lpout
      do 355 i=1,imjm
      ave3d(i,jj)= ave3d(i,jj)+rvor(i,j)
  355 continue
c
  350 continue
c
      numav= numav+1
c
      call dtgfix(idtgx,idtgx,24)
      newmon=idtgx(3:4)
c
      if(newmon.ne.month) then
      month= newmon
      c3ave= c3path(1:l3path)//month//c3av
      hccn= avpath(1:lpath)//month//hcc
      trpfil= avpath(1:lpath)//month//trpf
      c3grid= avpath(1:lpath)//month//c3g
```

```
      shist= avpath(1:lpath)//month//shst
      endif
c
  165 continue
c
      fac= 1.0/numav
      do 167 i=1,imjm
      ave2d(i,nfws)= ave2d(i,nfws)*fac
      ave2d(i,nfss)= ave2d(i,nfss)*fac
      ave2d(i,ndsw)= ave2d(i,ndsw)*fac
      ave2d(i,ncst)= ave2d(i,ncst)/dswiii(i)
      ave2d(i,nss)= ave2d(i,nss)*fac
      ave2d(i,ncss)= ave2d(i,ncss)/dswiii(i)
      ave2d(i,nudg)= ave2d(i,nudg)*fac
      ave2d(i,nvdg)= ave2d(i,nvdg)*fac
      ave2d(i,nulw)= ave2d(i,nulw)*fac
      ave2d(i,nrs)= ave2d(i,nrs)*fac
      ave2d(i,nclt)= ave2d(i,nclt)/olriii(i)
      ave2d(i,ncls)= ave2d(i,ncls)/olriii(i)
      ave2d(i,ntsf)= ave2d(i,ntsf)*fac
      ave2d(i,nqfl)= ave2d(i,nqfl)*fac
      ave2d(i,ncum)= ave2d(i,ncum)*fac
      ave2d(i,nlsp)= ave2d(i,nlsp)*fac
      ave2d(i,nps)= ave2d(i,nps)*fac
      ave2d(i,nrma)= ave2d(i,nrma)*fac
      ave2d(i,ntgf)= ave2d(i,ntgf)*fac
      ave2d(i,nrmg)= ave2d(i,nrmg)*fac
      dswiii(i)= dswiii(i)*fac
      olriii(i)= olriii(i)*fac
  167 continue
c
      temx= 3600.0*24.0/2.52e6
c
      do 260 i=1,imjm
      ave2d(i,nctt)= ave2d(i,nclt)+ave2d(i,ncst)
      ave2d(i,ncts)= ave2d(i,ncls)+ave2d(i,ncss)
      ave2d(i,ncsa)= ave2d(i,ncst)-ave2d(i,ncss)
      ave2d(i,ncla)= ave2d(i,nclt)-ave2d(i,ncls)
      ave2d(i,ncta)= ave2d(i,ncsa)+ave2d(i,ncla)
      ave2d(i,npcp)= ave2d(i,ncum)+ave2d(i,nlsp)
      ave2d(i,nevp)= temx*ave2d(i,nfws)
      ave2d(i,ntsf)= ave2d(i,ntsf)-273.15
      ave2d(i,ntgf)= ave2d(i,ntgf)-273.15
      ave2d(i,npme)= ave2d(i,npcp)-ave2d(i,nevp)
      ave2d(i,nvts)= fac*ave2d(i,nvts)-ave2d(i,ntsf)**2
      ave2d(i,nvtg)= fac*ave2d(i,nvtg)-ave2d(i,ntgf)**2
      ave2d(i,nsno)= ave2d(i,nsno)*fac
      ave2d(i,nwet)= ave2d(i,nwet)*fac
      ave2d(i,nwev)= fac*ave2d(i,nwev)-ave2d(i,nwet)**2
      ave2d(i,ntcc)= min(0.999,ave2d(i,ntcc)*fac)
      ave2d(i,nth2o)=ave2d(i,nth2o)*fac
      ave2d(i,nalb)= ave2d(i,nalb)*fac
      if(ave2d(i,nalb).gt.0.0)
     * ave2d(i,nalb)= 1.0-ave2d(i,ndsw)/ave2d(i,nalb)
      tmpp1(i)= tmpp1(i)*numav
  260 continue
c
      endif
c
      call s2pcs(im,jm,lm,tempcs,tmpp1,plog,cstt,pcsl,fac)
      call s2pcs(im,jm,lm,shtcs,shtcs(1,lm),plog,cssht,pcsl,fac)
      call s2pcs(im,jm,lm,rhcs,rhcs(1,lm),plog,csrh,pcsl,fac)
      call s2pcs(im,jm,lm,cldscs,cldscs(1,lm),plog,csclds,pcsl,fac)
      call s2pcs(im,jm,lm,uucs,uucs(1,lm),plog,csuu,pcsl,fac)
      call s2pcs(im,jm,lm,vvcs,vvcs(1,lm),plog,csvv,pcsl,fac)
      call s2pcs(im,jm,lm,strmcs,strmcs(1,lm),plog,csstrm,pcsl,fac)
c
```

```
      facx= 1.0/im
      do 262 j=1,jm
      ptzm(j)= 0.0
      do 264 i=1,im
      ptzm(j)= ptzm(j)+pt(i,j)
  264 continue
      ptzm(j)= ptzm(j)*facx
  262 continue
c
      call hadly(im,jm,lm,jtrun,mlmax,weight,poly,msort,csstrm)
c
      do 265 k=1,lm
      do 265 j=1,jm
      csclds(j,k)= max(0.0,csclds(j,k))
      csrh(j,k)= max(0.0,min(1.0,csrh(j,k)))
      csuu(j,k)= csuu(j,k)*rad/cosl(j)
      csvv(j,k)= csvv(j,k)*rad/cosl(j)
  265 continue
c
      if(lrec.eq.'CROSSECS') then
      do 362 i=1,7
      cap(i)(1:24)= llab(i+numave-7)
      cap(i)(25:64)='                                '
  362 continue
      capsun='AMIP TIMEAVE FROM '//idtgs//' TO '//idtge//'        '
      lencs= 7*lm*jm+lm+jm
      read(idtgs,'(2i2)')i1,i2
      itau= 100*i2+i1
      call dgwrit(jm,7,lencs,itau,idtgs,capsun,cap,cstt,'zmeancss')
      go to 120
      endif
c
c ********************************************************************
c
      do 15 kid=1,36
      if(lrec.eq.clrecs(kid)) then
      do 17 i=1,imjm
      avsum(i)= ave2d(i,kid)
   17 continue
c
      kl= kid
      klev=1
      iplev='    '
      go to 52
      endif
   15 continue
c
      do 57 kid=1,21*lpout
      if(lrec.eq.clrecl(kid)) then
      kl= 40+kid
      iplev= lrec(5:8)
      read(iplev,'(i4)') ip
      if(ip.gt.0) call levget(ip,pout,lpout,klev)
      print*, kid,lrec,klev
      do 59 i=1,imjm
      avsum(i)= ave3d(i,kid)*fac
   59 continue
      go to 52
      endif
   57 continue
c
      do 55 kid=1,2+14*lpout
c
      if(lrec.eq.clreca(kid)) then
c
      kl= 40+21*lpout+kid
      iplev= lrec(5:8)
```

```
      read(iplev,'(i4)') ip
      if(ip.gt.0) call levget(ip,pout,lpout,klev)
      print*, kid,lrec,klev
c
c  sea level pressure
c
      if(lrec.eq.lslp) then
      call ave(imjm,fac,slp,avsum)
      klev= 1
c
      else if(lrec.eq.lslpv) then
      call var(imjm,fac,slp,slpv,avsum)
      klev= 1
c
c  temperature
c
      else if(lrec.eq.ltmp(klev)) then
      call ave(imjm,fac,temps(1,klev),avsum)
c
      else if(lrec.eq.ltmpv(klev)) then
      call var(imjm,fac,temps(1,klev),tempsv(1,klev),avsum)
c
c  geopotential
c
      else if(lrec.eq.lhgt(klev)) then
      call ave(imjm,fac,phips(1,klev),avsum)
c
      else if(lrec.eq.lhgtv(klev)) then
      call var(imjm,fac,phips(1,klev),phipsv(1,klev),avsum)
c
c  specific humidity
c
      else if(lrec.eq.lsht(klev)) then
      call ave(imjm,fac,sphum(1,klev),avsum)
c
      else if(lrec.eq.lshtv(klev)) then
      call var(imjm,fac,sphum(1,klev),sphumv(1,klev),avsum)
c
c  u component wind
c
      else if(lrec.eq.luwn(klev)) then
      call ave(imjm,fac,upres(1,klev),avsum)
c
      else if(lrec.eq.luwnv(klev)) then
      call var(imjm,fac,upres(1,klev),upresv(1,klev),avsum)
      do 210 i=1,imjm
      avsum(i)= avsum(i)*radsq*ocos(i)
  210 continue
c
c  v component wind
c
      else if(lrec.eq.lvwn(klev)) then
      call ave(imjm,fac,vpres(1,klev),avsum)
c
      else if(lrec.eq.lvwnv(klev)) then
      call var(imjm,fac,vpres(1,klev),vpresv(1,klev),avsum)
      do 220 i=1,imjm
      avsum(i)= avsum(i)*radsq*ocos(i)
  220 continue
c
c  stream function
c
      else if(lrec.eq.lpsi(klev)) then
      call ave(imjm,fac,psi(1,klev),avsum)
c
      else if(lrec.eq.lpsiv(klev)) then
      call var(imjm,fac,psi(1,klev),psiv(1,klev),avsum)
```

```
c
c  velocity potential
c
      else if(lrec.eq.lchi(klev)) then
      call ave(imjm,fac,chi(1,klev),avsum)
c
      else if(lrec.eq.lchiv(klev)) then
      call var(imjm,fac,chi(1,klev),chiv(1,klev),avsum)
c
      endif
      go to 52
      endif
   55 continue
c
      go to 120
c
   52 continue
c
      call sumcos(im,jm,cosl,avsum,gmean)
      print*,' gmean= ',gmean
c
      ci= plotc(1,kl)
      co= plotc(2,kl)
      add=plotc(3,kl)
      amp=plotc(4,kl)
c
      call labls(kl,lrec(1:3),iplev,isize,idtg(1),idtgs,idtge,gmean
     *, capsun(1:24),ci,co,add,amp,llab(kl),ident)
      print*, ident
c
      if(superg) then
      imax= 288
      jmax= 145
      else
      imax= 144
      jmax= 73
      endif
c
      ik= 1
      if(isize.ne.iszold) ik= 0
      call hterp(ik,avsum,im,jm,glob,xin,yin,imax,jmax,alat,vwk)
      if(lrec(1:3).eq.luwn(klev)(1:3)) call wndfix(glob,imax,jmax)
      if(lrec(1:3).eq.lvwn(klev)(1:3)) call wndfix(glob,imax,jmax)
      if(plotc(5,kl).eq.1) call fixit(glob,imax*jmax,0.0)
      call qprnth(glob,ident,1,1,imax,jmax)
      call qpnh(glob,ident,1,1,imax,jmax)
c
      lenx= imax*jmax
      if(superg) then
c     call ritasc(ident,lenx,glob)
      call cfwrit(filarg,ident,glob,lenx,0,istat)
      else
      call gloflp(imax,jmax,glob,gflip)
c     call ritasc(ident,lenx,gflip)
      call cfwrit(filsml,ident,gflip,lenx,0,istat)
      endif
c
      go to 120
c
  160 continue
      stop
      end
      subroutine fixdat(kk,tt,mtt,imjm,lm)
c
      dimension mtt(imjm,lm),tt(imjm,lm)
c
      icount= 0
```

```
c
      do 10 k=1,lm
      do 10 i=1,imjm
      ll=ibits(mtt(i,k),0,24)
      if(ll.ne.0) then
      tt(i,k)= 0.0
      icount= icount+1
      endif
   10 continue
      if(icount.eq.0) return
      print*,kk,' fixdat found ',icount,' bad values'
      return
      end
      subroutine ave(imjm,fac,x,avsum)
c
      dimension x(imjm),avsum(imjm)
c
      do 10 i=1,imjm
      avsum(i)= fac*x(i)
   10 continue
      return
      end
      subroutine avevar(imjm,ll,xl,ave,var)
c
      dimension ave(imjm,ll),var(imjm,ll),xl(imjm,ll)
c
      do 10 k=1,ll
      do 10 i=1,imjm
      ave(i,k)= ave(i,k)+xl(i,k)
      var(i,k)= var(i,k)+xl(i,k)**2
   10 continue
c
      return
      end
      subroutine bcubv(iderv,f,ix,jy,dout,mn,yr,yin,ten)
c
c  parameter list
c
c  f,ix,jy,dout,mn,xin,yin - see below
c
      dimension f(ix,jy),dout(mn),yr(ix,jy),yin(mn)
c
      dimension fxx(ix,jy),fyy(ix,jy),pjy(mn,4),tp1(mn,4)
      dimension ipt(mn),jpt(mn)
c
c
c          a bicubic spline interpolator to interpolate from a grid
c          with constant i (first dimension) grid spacing and variable
c          j (second dimension) grid spacing t0 a grid with
c          variable grid spacing. all grids are assumed to have point
c          (i,1) in the lower left corner with i increasing to the right
c          and j increasing upward.
c
c
c          compute ipt,jpt and pjy
c
      call setupv(iderv,yr,yin,mn,ix,jy,pjy,ipt,jpt,tp1)
      jm= mn/ix
      ixm1=ix-1
      jym1=jy-1
      jym2=jy-2
      ijm2=ix*jy-2
      ixjym2=ix*jym2
      mn4=mn*4
c
c
c          compute fyy
```

```
c
      ll= ixjym2+ix
      do 100 i=1,ll
      fxx(i,2)= yr(i,2)-yr(i,1)
  100 continue
c
      do 210 i=1,ixjym2
      fyy(i,2)= ten*(fxx(i,3)*(f(i,1)-f(i,2))+fxx(i,2)*(f(i,3)-f(i,2)))
     * /(fxx(i,3)*fxx(i,3)*fxx(i,2))
      fxx(i,2)= fxx(i,2)/fxx(i,3)
  210 continue
c
      call trdivv(ix,jym2,fxx(1,2),fyy(1,2))
c
      call zilch(fyy,ix)
      call zilch(fyy(1,jy),ix)
c
      call gathv(mn,ix,jy,ipt,jpt,fyy,f,tp1)
c
      do 130 i=1,mn4
      tp1(i,1)=tp1(i,1)*pjy(i,1)
  130 continue
      do 140 i=1,mn
      dout(i)=tp1(i,1)+tp1(i,2)+tp1(i,3)+tp1(i,4)
  140 continue
      return
      end
      subroutine dgwrit(id1,id2,len,itau,cdtg,capsun,cap,x,label)
c
      dimension x(len)
      character*8 cdtg
      character*8 label
      character*48 filedg
      character*14 cform
      character*64 cap(id2),capsun
c
      character*32 dagdir
c
      dagdir='/u/b/rosmond/pcmdi/pcmflds/'
      call chlen(dagdir,lendir)
      write(cform,90) label,itau
   90 format(a8,i6.6)
c
        filedg= dagdir(1:lendir) // cform
c
      read(cdtg,'(f2.0,f6.0)') yr, xdtg
c
c     open(unit=44,file=filedg,form='unformatted')
      open(unit=44,file=filedg,form='formatted')
c
      rd1= id1
      rd2= id2
      rlen= len
  143 format(a64)
      write(44,144) rd1,rd2,rlen,yr,xdtg
      write(44,143) capsun
      write(44,143) (cap(i),i=1,id2)
      write(44,144) (x(i),i=1,len)
  144 format(5e16.9)
      close(unit=44)
      print*,' writing ',filedg
      print*, capsun
      do 200 k=1,id2
      print*,k,cap(k)
  200 continue
c
      return
```

```
      end
      subroutine fixit(a,len,alow)
c
c  this routine is used to adjust certain variables which during
c  interpolation are given unrealistic values, e.g., negative vapor
c  pressures.
c
c  ***************************************************************************
c
c
      implicit real (a-h,o-z)
c
      real a(len)
c
      if(alow.ge.0.0) then
      do 10 i=1,len
      a(i)= max(a(i),alow)
   10 continue
      else
      tem= -alow
      do 20 i=1,len
      a(i)= min(a(i),tem)
   20 continue
      endif
      return
      end
      subroutine geotrp(im,jm,lm,lpout,psout,tt,phi,sgeo,slp
     *, plog,temps,phips)
c
c  this routine reads in the model forecasts of geopotential and
c  processes them so they can be output as standard height fields.
c
c  it also processes the temperature output fields and outputs
c  these as standard fields
c
      dimension plog(im*jm,lm+1),phips(im*jm,lpout)
     *, temps(im*jm,lpout),phi(im*jm,lm),tt(im*jm,lm+1)
     *, phibot(im*jm),phisuf(im*jm),sgeo(im*jm),slp(im*jm)
     *, psout(lpout)
c
c  convert geopotentials to d-values at sigma pressures
c
      ograv= 1.0/9.80616
      do 11 k=1,lm
      do 11 i=1,im*jm
      phi(i,k)= phi(i,k)*ograv
      tt(i,k)= tt(i,k)-273.15
   11 continue
c
      p0log= log(1000.0)
      do 13 i=1,im*jm
      phibot(i)= 0.0
      slplog= log(slp(i))
      plog(i,lm+1)= max(slplog,p0log)+0.0001
   13 continue
c
      call s2ptrp(im,jm,lm,lpout,phi,phibot,plog,phips,psout,0.5)
c
      call s2ptrp(im,jm,lm,lpout,tt,tt(1,lm+1),plog,temps,psout,0.5)
c
      return
      end
      subroutine gloflp(imax,jmax,glob,gflip)
      dimension glob(imax,jmax),gflip(jmax,imax)
c
      do 10 j=1,jmax
      do 10 i=1,imax
```

```
      gflip(j,i)= glob(i,jmax+1-j)
   10 continue
      return
      end
      subroutine hadly(im,jm,lm,jtrun,mlmax,weight,poly
     *, msort,stream)
c
      dimension meq0(jtrun),msort(mlmax),spcoef(mlmax,2)
     *, poly(mlmax*2,jm),weight(jm),work4(jtrun)
     *, work5(im,jm),stream(jm,lm)
c
       m= 0
       do 150 ml=1,mlmax
       if(msort(ml).ne.1) go to 150
       m= m+1
       meq0(m)= ml
        work4(m)= m/sqrt(4.0*m*m-1.0)
  150  continue
c
      do 450 k=1,lm
c
      do 452 j=1,jm
      do 452 i=1,im
      work5(i,j)= stream(j,k)
  452 continue
c
        call tranrs(0,1,poly,weight,work5,spcoef)
c
        work5(1,1)=0.0
        work5(2,1)=0.5*spcoef(meq0(3),1)*work4(2)
        jtrun2=jtrun-2
c
        do 446 mm=2,jtrun2
        ml= meq0(mm)
        ml2= meq0(mm+2)
        work5(mm+1,1)=(mm*spcoef(ml2,1)*work4(mm+1)-(mm+1)*work4(mm)
     *  *spcoef(ml,1))/(mm*(mm+1))
  446   continue
c
        ml1= meq0(jtrun-1)
        work5(jtrun,1)=-spcoef(ml1,1)*work4(jtrun)/(jtrun-1.0)
      call zilch(spcoef,mlmax*2)
c
      do 447 l=2,jtrun
      spcoef(meq0(l),1)= work5(l,1)
  447 continue
c
       call transr(1,poly,spcoef,work5,0,0,0)
c
       do 460 j=1,jm
       stream(j,k)= work5(1,j)
  460 continue
c
  450 continue
c
      return
      end
      subroutine hterp(ik,z,im,jm,glob,xin,yin,ix,jy,alat,v)
c
      dimension glob(ix,jy),w1(im,jm+2)
     *, alat(jm+2),xin(ix,jy),yin(ix,jy)
c
      real v(1000)
c
      dimension z(im,jm)
c
      save lenx
```

```
c
c
      data  rad/6.375e6/
c
      pi= 4.0*atan(1.0)
      pio2= pi*0.5
      if(ik.eq.0) then
      lenx= ix*jy
      tem= im/float(ix)
      do 10 i=2,ix
      xin(i,1)= 1.0+tem*(i-1.0)
   10 continue
      xin(1,1)= 1.00001
      do 15 j=2,jy
      do 15 i=1,ix
   15 xin(i,j)= xin(i,1)
c
      jym= jy-1
      do 17 j=2,jym
      tem= -pio2+(j-1.0)*pi/jym
      do 17 i=1,ix
   17 yin(i,j)= tem
      do 12 i=1,ix
      yin(i,1)= -pio2
      yin(i,jy)= pio2
   12 continue
      endif
c
      do 22 i=1,im*jm
   22 w1(i,2)= z(i,1)
      call sumit(im,z,ps)
      call sumit(im,z(1,jm),pn)
      do 25 i=1,im
      w1(i,1)= ps/im
   25 w1(i,jm+2)= pn/im
      call bcubiy(ik,2,w1,im,jm+2,glob,lenx,alat,xin,yin,v,1.0,1.0)
c
      return
      end
      subroutine labls(kid,lrec,iplev,isize,idtg,idtgs,idtge,gmean
     *, capsun,ci,co,add,amp,llab,ident)
c
      character*192 ident
      character*8 idtg,idtgs,idtge
      character*24 capsun
      character*4 iplev,ctau
      character*1 iflap
      character*3 lrec
c
      character*24 llab
c
      k1= 1
      do 12 k=1,24
      ident(k1:k1+7)='        '
      k1= k1+8
   12 continue
c
      ident(169:192)= capsun
      if(iplev.eq.'    ') then
      ident(49:72)= llab
      ident(73:88)= '                '
      ctau= '   0'
      else
      write(ident(49:88),100) llab,iplev
  100 format(a24,' AT ',a4,' MBS')
      ctau= iplev
      endif
```

```
c
      write(ident(89:136),200) idtgs,idtge,gmean
  200 format('TIMEAVE FROM ',A8,' TO ',A8,' mean=',f9.3)
c
      iflap= '$'
      lenx= 10536
      if(isize.gt.0) lenx= 24+288*145
      if(isize.gt.0) iflap='N'
      ident(1:16)=lrec // iflap // ctau  // idtgs
      ident(48:48)='0'
      write(ident(17:24),210) lenx
  210 format(2x,i6)
      ident(33:40)=' TIMEAVE'
c
      if(ci.lt.0.01) then
      write(ident(137:168),'(4f8.4)') ci,co,add,amp
c
      else if(abs(add).gt.100.0) then
      write(ident(137:168),'(4f8.1)') ci,co,add,amp
c
      else
      write(ident(137:168),'(4f8.3)') ci,co,add,amp
c
      endif
c
      print 300, ident
  300 format(24a8)
      return
      end
      subroutine levget(ip,pout,lpout,klev)
      dimension pout(lpout)
c
      klev= 0
      do 10 k=1,lpout
      if(ip.eq.int(pout(k))) then
      klev= k
      return
      endif
   10 continue
      return
      end
      subroutine p2kapa(imjm,lm,asig,bsig,pt,psigc,plog,pk2,pll)
c
      dimension asig(lm+1),bsig(lm+1),pt(imjm),pk2(imjm,lm)
     *, psigc(imjm,lm),plog(imjm,lm),pll(imjm,lm)
c
c  compute pressures on the sigma surfaces of the forecast model
c  and raise to the capa power.
c
      dcapa= 3.5
      capa= 1.0/3.5
      capap1= 1.0+capa
      p0= 1000.0
      opok= 1.0/p0**capa
      p0log= log(p0)
      tem1= opok
c
      px= 1.0/capap1
      do 19 i= 1,imjm
      pwrk2= asig(2)+bsig(2)*pt(i)
      pk2(i,1)= opok*pwrk2**capa
      psigc(i,1)= (pk2(i,1)*pwrk2-tem1)*px/(pwrk2-1.0)
      plog(i,1)= p0log+dcapa*log(psigc(i,1))
      pll(i,1)= exp(plog(i,1))
   19 continue
c
      do 28 k=2,lm
```

```
      do 27 i=1,imjm
      pwrk1= asig(k)+bsig(k)*pt(i)
      pwrk2= asig(k+1)+bsig(k+1)*pt(i)
      pk2(i,k)= opok*pwrk2**capa
      psigc(i,k)=(pk2(i,k)*pwrk2-pk2(i,k-1)*pwrk1)*px
     * /(pwrk2-pwrk1)
      plog(i,k)= p0log+dcapa*log(psigc(i,k))
      pll(i,k)= exp(plog(i,k))
   27 continue
   28 continue
c
      return
      end
      subroutine ritasc(ihdg,len,x)
      dimension x(len)
      logical first
c
      character*192 ihdg
      data first/.true./
c
      if(first) then
      first=.false.
      open(unit=16,file='/scr/rosmond/pcmdi/pcmflds/asfile'
     * ,form='formatted')
      endif
c
      write(16,100) len,ihdg
      write(16,200) (x(i),i=1,len)
  100 format(i8,a192)
  200 format(8e10.4)
      return
      end
      subroutine s2pcs(im,jm,lm,y,boty,pin,dout,pout,fac)
c
      dimension y(im,jm,lm+1),pout(lm),boty(im,jm),dout(jm,lm)
     *, pin(im,jm,lm+1)
c
      dimension yh(im,lm+1),pinh(im,lm+1),doh(im,lm)
     *, poh(im,lm)
c
      eps= 1.0e-6
      ten= 0.5
      imlm= im*lm
      do 5 k=1,lm
      do 5 i=1,im
      poh(i,k)= pout(k)
    5 continue
      do 1 j=1,jm
      do 4 i=1,im
      yh(i,lm+1)= boty(i,j)
    4 continue
      do 2 k=1,lm
      do 2 i=1,im
      yh(i,k)= y(i,j,k)
    2 continue
      do 3 k=1,lm+1
      do 3 i=1,im
    3 pinh(i,k)= pin(i,j,k)
      call bcubv(0,yh,im,lm+1,doh,imlm,pinh,poh,ten)
      xfac= fac/im
      do 6 k=1,lm
      dout(j,k)= 0.0
      do 6 i=1,im
      dout(j,k)= dout(j,k)+doh(i,k)*xfac
    6 continue
    1 continue
      return
```

```
      end
      subroutine s2ptrp(im,jm,lm,lpout,y,boty,pin,dout,pout,ten)
c
      dimension y(im,jm,lm+1),pout(lpout),boty(im,jm),dout(im,jm,lpout)
     *, pin(im,jm,lm+1)
c
      dimension yh(im,lm+1),pinh(im,lm+1),doh(im,lpout)
     *, poh(im,lpout)
c
      eps= 1.0e-6
      imlm= im*lpout
      do 5 k=1,lpout
      do 5 i=1,im
      poh(i,k)= pout(k)
    5 continue
      do 1 j=1,jm
      do 4 i=1,im
      yh(i,lm+1)= boty(i,j)
    4 continue
      do 2 k=1,lm
      do 2 i=1,im
      yh(i,k)= y(i,j,k)
    2 continue
      do 3 k=1,lm+1
      do 3 i=1,im
    3 pinh(i,k)= pin(i,j,k)
      call bcubv(0,yh,im,lm+1,doh,imlm,pinh,poh,ten)
      do 6 k=1,lpout
      do 6 i=1,im
      dout(i,j,k)= doh(i,k)
    6 continue
    1 continue
      return
      end
      subroutine sdet
c
      parameter (im=144, jm=im/2, lm= 18, lpout=6)
      parameter (jtrun= 2*((1+(im-1)/3)/2), mlmax= (jtrun+1)*jtrun/2)
      parameter (idim2= 2*mlmax)
      parameter (nspec=4*lm+1)
      parameter (numave= 40+21*lpout+2+14*lpout+7)
      parameter (imjm=im*jm)
c
c
      parameter (lccn= 350+lm*(12+5*lm))
c
c  ********************************************************************
c
      dimension ccn(lccn)
      equivalence (c,ccn)
      character*8 idtg
c
      common/cnstnt/ c(300),itau,ithist,idtg(3),sarray(lm,lm)
     *, carray(lm,lm),evecin(lm,lm),evectr(lm,lm),tmcor(lm,lm)
     *, eigval(lm),spalm(lm),pmcor(lm),pbcor(lm),tmean(lm)
     *, asig(lm+1),bsig(lm+1),dasig(lm),dbsig(lm)
     *, sige(lm+1),dsig(lm),pkfac(lm)
     *, ptmean,ccpad(100)
c
c  ********************************************************************
c
      equivalence
     1  (c(1),jmm1),       (c(3),jmm2),       (c(4),lncrec)
     2, (c(5),iday),       (c(6),tofday)
     3, (c(7),itauo),      (c(8),julian),     (c(9),jskip)
     4, (c(10),jsplit),    (c(11),taui),      (c(12),dtrad)
     5, (c(13),taue),      (c(14),tice),      (c(15),forwrd)
```

```
     6, (c(16),cosd),      (c(17),ddraft),   (c(18),jtrunp)
     7, (c(19),eccld),     (c(20),rbear),    (c(21),prntij)
     8, (c(22),prevap),    (c(23),radsq)
     9, (c(25),omega),     (c(26),tauave),   (c(27),pi)
     1, (c(28),pi2),       (c(29),rgas),     (c(30),capa)
     2, (c(31),cp),        (c(32),hltm),     (c(33),fim)
     3, (c(41),fluxcl),    (c(42),isunfx),   (c(43),ptop)
     4, (c(44),tauh),      (c(45),krnit),    (c(46),ls)
     5, (c(47),mombdg),    (c(48),evaprh),   (c(49),vis)
     6, (c(50),p00),       (c(51),dta),      (c(52),ops)
     7, (c(53),hybrd),     (c(54),ksgeo),    (c(55),pok)
     8, (c(56),dgtime),    (c(57),ecadv)
c
        equivalence
     1  (c(68),taud),      (c(69),dt)
     2, (c(70),pcon),      (c(71),pconkt),   (c(72),ls1)
     3, (c(73),skew),      (c(74),grav2),    (c(75),delmf)
     4, (c(76),ithfg),     (c(77),itaufg),   (c(78),lsx)
     5, (c(79),tseres),    (c(80),prftop),   (c(81),dtgpcm)
     6, (c(93),daypyr),    (c(94),day),      (c(95),rotper)
     7, (c(96),iqnx),      (c(97),perhl),    (c(98),decmax)
     8, (c(99),eccn),      (c(100),rad),     (c(101),grav)
     9, (c(102),critl),    (c(103),p608),    (c(104),lmid)
     1, (c(105),pcp),      (c(106),frqdib),  (c(107),cdtgoi)
     2, (c(109),ipfcst),   (c(110),ipsigp),  (c(111),ipinit)
     3, (c(112),ipoutp),   (c(113),tau),     (c(114),drgdry)
     4, (c(115),icnmax),   (c(116),frqlim),  (c(117),beta)
     5, (c(118),nmodev),   (c(119),dinit),   (c(120),taudb)
     6, (c(121),uvmax),    (c(123),facrad),  (c(134),gamfac)
     8, (c(135),nrnt),     (c(136),capap1),  (c(137),cuprh)
c
        equivalence
     1  (c(138),novar),    (c(139),cdet),    (c(140),dcapa)
     2, (c(141),nozone),   (c(142),sstclm),  (c(143),updayc)
     3, (c(144),incrup),   (c(145),incwnd),  (c(146),pcmdi)
     4, (c(151),nsdedy),   (c(152),kmonth),  (c(153),mnthdy)
     5,                    (c(157),rsdist),  (c(158),sind)
c
c  ************************************************************************
c
c
      logical first
c
      save sob,perhlr,reccn,h,maxday,y,iiday
c
      data first/.true./
      f(x) = reccn*(1.-eccn*cos(x-perhlr))**2
c
      if(first) then
      iqnx= 80
      pib180 = pi/180.
      sob    = sin(decmax*pib180)
      perhlr = perhl * pib180
      reccn  = 1./(1.-eccn*eccn)**1.5
      h      = pi2/daypyr
      maxday = daypyr + .1
      endif
c
c     set the date
c
      call daynum(idtg(2),idy,kmonth,mnthdy,mh,nsdedy,mhy)
c
c     integrate for the earth's position
c
      if (.not.first .and. nsdedy .ne. iqnx) go to 105
      first = .false.
      y = 0.
```

```
      iiday = nsdedy - iqnx
      if (iiday .eq. 0) go to 110
      if (iiday .lt. 0) iiday = iiday + maxday
100   iiday = iiday - 1
105   t1 = f(y)*h
      t2 = f(y+t1*.5)*h
      t3 = f(y+t2*.5)*h
      t4 = f(y+t3)*h
      y = y + (t1+2.*(t2+t3)+t4)/6.
      if (iiday .gt. 0) go to 100
c
c     compute declination and earth-sun distance
c
110   sind   = sob*sin(y)
      cosd   = sqrt(1.-sind*sind)
      rsdist = ((1.-eccn*eccn)/(1.-eccn*cos(y-perhlr)))**2
c
      print 120, nsdedy,kmonth,mnthdy,sind
  120 format(' nsdedy=',i3,', month=',i2,', day=',i2
     *,', sine of solar declination=',f6.4)
c
      return
      end
      subroutine solar(s0cosz,idtgx,sinl,cosl)
c
      parameter (im=144, jm=im/2, lm= 18, lpout=6)
      parameter (jtrun= 2*((1+(im-1)/3)/2), mlmax= (jtrun+1)*jtrun/2)
      parameter (idim2= 2*mlmax)
      parameter (nspec=4*lm+1)
      parameter (numave= 40+21*lpout+2+14*lpout+7)
      parameter (imjm=im*jm)
c
c
      parameter (lccn= 350+lm*(12+5*lm))
c
c  ********************************************************************
c
      dimension ccn(lccn)
      equivalence (c,ccn)
      character*8 idtg
c
      common/cnstnt/ c(300),itau,ithist,idtg(3),sarray(lm,lm)
     *, carray(lm,lm),evecin(lm,lm),evectr(lm,lm),tmcor(lm,lm)
     *, eigval(lm),spalm(lm),pmcor(lm),pbcor(lm),tmean(lm)
     *, asig(lm+1),bsig(lm+1),dasig(lm),dbsig(lm)
     *, sige(lm+1),dsig(lm),pkfac(lm)
     *, ptmean,ccpad(100)
c
c  ********************************************************************
c
      equivalence
     1  (c(1),jmm1),       (c(3),jmm2),       (c(4),lncrec)
     2, (c(5),iday),       (c(6),tofday)
     3, (c(7),itauo),      (c(8),julian),     (c(9),jskip)
     4, (c(10),jsplit),    (c(11),taui),      (c(12),dtrad)
     5, (c(13),taue),      (c(14),tice),      (c(15),forwrd)
     6, (c(16),cosd),      (c(17),ddraft),    (c(18),jtrunp)
     7, (c(19),eccld),     (c(20),rbear),     (c(21),prntij)
     8, (c(22),prevap),    (c(23),radsq)
     9, (c(25),omega),     (c(26),tauave),    (c(27),pi)
     1, (c(28),pi2),       (c(29),rgas),      (c(30),capa)
     2, (c(31),cp),        (c(32),hltm),      (c(33),fim)
     3, (c(41),fluxcl),    (c(42),isunfx),    (c(43),ptop)
     4, (c(44),tauh),      (c(45),krnit),     (c(46),ls)
     5, (c(47),mombdg),    (c(48),evaprh),    (c(49),vis)
     6, (c(50),p00),       (c(51),dta),       (c(52),ops)
     7, (c(53),hybrd),     (c(54),ksgeo),     (c(55),pok)
```

```
     8, (c(56),dgtime),    (c(57),ecadv)
c
        equivalence
     1  (c(68),taud),      (c(69),dt)
     2, (c(70),pcon),      (c(71),pconkt),   (c(72),ls1)
     3, (c(73),skew),      (c(74),grav2),    (c(75),delmf)
     4, (c(76),ithfg),     (c(77),itaufg),   (c(78),lsx)
     5, (c(79),tseres),    (c(80),prftop),   (c(81),dtgpcm)
     6, (c(93),daypyr),    (c(94),day),      (c(95),rotper)
     7, (c(96),iqnx),      (c(97),perhl),    (c(98),decmax)
     8, (c(99),eccn),      (c(100),rad),     (c(101),grav)
     9, (c(102),critl),    (c(103),p608),    (c(104),lmid)
     1, (c(105),pcp),      (c(106),frqdib),  (c(107),cdtgoi)
     2, (c(109),ipfcst),   (c(110),ipsigp),  (c(111),ipinit)
     3, (c(112),ipoutp),   (c(113),tau),     (c(114),drgdry)
     4, (c(115),icnmax),   (c(116),frqlim),  (c(117),beta)
     5, (c(118),nmodev),   (c(119),dinit),   (c(120),taudb)
     6, (c(121),uvmax),    (c(123),facrad),  (c(134),gamfac)
     8, (c(135),prnt),     (c(136),capap1),  (c(137),cuprh)
c
        equivalence
     1  (c(138),novar),    (c(139),cdet),    (c(140),dcapa)
     2, (c(141),nozone),   (c(142),sstclm),  (c(143),updayc)
     3, (c(144),incrup),   (c(145),incwnd),  (c(146),pcmdi)
     4, (c(151),nsdedy),   (c(152),kmonth),  (c(153),mnthdy)
     5,                    (c(157),rsdist),  (c(158),sind)
c
c  ***************************************************************************
c
c
      character*8 idtgx
      dimension s0cosz(imjm),sinl(jm),cosl(jm)
c
      data solcst/1365.0/, pi/3.1415926/
c
      idtg(2)= idtgx
c
c  compute siderial day and solar declination for this day
c
      call sdet
c
c  adjust solar constant for this day; see Paltridge and Platt (1976)
c  use value at 1200 for daily mean
c
      fday= nsdedy-0.5
      tho= min(2.0*pi,2.0*pi*fday/365.0)
      tem2= 2.0*tho
      sin0= sin(tho)
      cos0= cos(tho)
      sin20= sin(tem2)
      cos20= cos(tem2)
      s0= solcst*(1.000110+0.034221*cos0+0.001280*sin0
     * +0.000719*cos20+0.000077*sin20)
c
c integral of cos(Z) over daylight period, multiplied by s0
c
      do 10 i=1,imjm
      j= 1+(i-1)/im
      a= sinl(j)*sind
      b= cosl(j)*cosd
      arg= a/b
      if(abs(arg).le.1.0) then
      h0= acos(-arg)
      s0cosz(i)= a+b/h0*sin(h0)
      s0cosz(i)= s0cosz(i)*h0/pi
      s0cosz(i)= s0*s0cosz(i)
      else if(arg.gt.1.0) then
```

```
      s0cosz(i)= s0*a
      else if(arg.lt.-1.0) then
      s0cosz(i)= 0.0
      endif
c     if(mod(i,144).eq.0) print 100, j,a,b,arg,h0,s0cosz(i)
c 100 format(1x,'j=',i3,' a=',e10.4,' b=',e10.4,' arg=',e10.4
c    *,' h0=',e10.4,' s0cosz=',e10.4)
   10 continue
      return
      end
      subroutine sumcos(im,jm,cosl,x,gmean)
c
      dimension cosl(jm),x(im,jm)
      logical first
c
      save denom
c
      data first/.true./
      if(first) then
      denom= 0.0
      do 1 j=1,jm
    1 denom= denom+cosl(j)
      denom= 1.0/(im*denom)
      first=.false.
      endif
c
      gmean= 0.0
      do 2 j=1,jm
      do 2 i=1,im
    2 gmean= gmean+cosl(j)*x(i,j)
      gmean= gmean*denom
      return
      end
      subroutine svar(imjm,lm,kmid,pt,sgeo,pdiff,tsave,phi,tt,tmpp1,slp)
c
c  subroutine to process single level surface variable output
c
c  this routine reads the model fields of pbl variables and ground
c  hydrology variables and processes them so they can be output as
c  standard fields.
c
      dimension sgeo(imjm),pt(imjm),pdiff(imjm),tsave(imjm)
     *, tt(imjm,lm+1),tmpp1(imjm),phi(imjm,lm),slp(imjm)
      dimension ppp(imjm)
c
c delta p method for finding sea level pressure
c
c  add pdiff to terrain pressure to get forecast sea level pressure
c  compute correction to pdiff due to change in temperature in
c  bottom half of atmosphere during forecast
c
      cp= 1005.45
      rgas= cp/3.5
      temx= 1.0/(rgas*log(10.0))
      ocp= 1.0/cp
c
c  compute t1000 from deep layer thickness change and adiabatically
c  from bottom model level pressure.
c  we constrain the 1000 mb temperature to be between isothermal
c  and adiabatic w.r.t. the bottom model level temperature
c
      do 40 i=1,imjm
      ppp(i)= temx*(phi(i,kmid)-sgeo(i))-tsave(i)
      p2= tmpp1(i)+ppp(i)
      p1= tt(i,lm)+ocp*sgeo(i)
      p2= min(p2,p1)
      p2= max(p2,tt(i,lm))
```

```
c
c  adjust pdiff due to change in subterrain temperature
c
      p1= p2-tmpp1(i)
      tt(i,lm+1)= p2
      p2= rgas*tt(i,lm+1)*tt(i,lm+1)
      p2= sgeo(i)*(pt(i)+pdiff(i))/p2
      ppp(i)= p1*p2
      slp(i)= pt(i)+pdiff(i)-p1*p2
   40 continue
c
      return
      end
      subroutine trngra(ll,cim,poly,dpoly,s,dlpl,dtpl)
c
      parameter (im=144, lm=18, lpout=6)
      parameter (jm= im/2, im2= im/2,  jm2= jm/2)
      parameter (jtrun= 2*((1+(im-1)/3)/2),  mlmax= (jtrun+1)*jtrun/2)
      parameter (imlm=im*lm,  imjm=im*jm,  idim2= mlmax*2)
c
      parameter (mlx= (jtrun/2)*((jtrun+1)/2))
      parameter (jump= im+3)
c
      common/ fftcom/ trigs(im,2),ifax(19)
c
      dimension cim(mlmax),poly(idim2,jm2),dpoly(idim2,jm2)
     *, s(mlmax,2,lm),dlpl(im,jm,lm),dtpl(im,jm,lm)
c
      dimension cu(jump,jm),cv(jump,jm),work(imjm,2)
c
      do 15 k=1,ll
      do 27 m= 1,jump*jm2
      cu(m,1)= 0.0
      cu(m,jm2+1)= 0.0
      cv(m,1)= 0.0
      cv(m,jm2+1)= 0.0
   27 continue
c
      do 20 j=1,jm2
      jj= jm+1-j
c
      ml= 2*mlx
CDIR@ IVDEP
      do 50 m=2,jtrun,2
      ml= ml+1
      mm= 2*m-1
      mp= mm+1
      cu(mm,j)=         +cim(ml)*poly(ml,j)*s(ml,2,k)
c
      cu(mm,jj)=        +cim(ml)*poly(ml,j)*s(ml,2,k)
c
      cu(mp,j)=         -cim(ml)*poly(ml,j)*s(ml,1,k)
c
      cu(mp,jj)=        -cim(ml)*poly(ml,j)*s(ml,1,k)
c
      cv(mm,j)=         -dpoly(ml,j)*s(ml,1,k)
c
      cv(mm,jj)=        +dpoly(ml,j)*s(ml,1,k)
c
      cv(mp,j)=         -dpoly(ml,j)*s(ml,2,k)
c
      cv(mp,jj)=        +dpoly(ml,j)*s(ml,2,k)
c
   50 continue
c
      m1= 0
      do 30 l=jtrun-1,1,-2
```

```
CDIR@ IVDEP
      do 40 m=1,l
      mm= 2*m-1
      mp= mm+1
      ml= m+m1
      mk= ml+mlx
c
      cu(mm,j)= cu(mm,j)+cim(ml)*poly(ml,j)*s(ml,2,k)
     *                  +cim(mk)*poly(mk,j)*s(mk,2,k)
c
      cu(mm,jj)= cu(mm,jj)+cim(ml)*poly(ml,j)*s(ml,2,k)
     *                  -cim(mk)*poly(mk,j)*s(mk,2,k)
c
      cu(mp,j)= cu(mp,j)-cim(ml)*poly(ml,j)*s(ml,1,k)
     *                  -cim(mk)*poly(mk,j)*s(mk,1,k)
c
      cu(mp,jj)= cu(mp,jj)-cim(ml)*poly(ml,j)*s(ml,1,k)
     *                  +cim(mk)*poly(mk,j)*s(mk,1,k)
c
      cv(mm,j)= cv(mm,j)-dpoly(ml,j)*s(ml,1,k)
     *                  -dpoly(mk,j)*s(mk,1,k)
c
      cv(mm,jj)= cv(mm,jj)+dpoly(ml,j)*s(ml,1,k)
     *                  -dpoly(mk,j)*s(mk,1,k)
c
      cv(mp,j)= cv(mp,j)-dpoly(ml,j)*s(ml,2,k)
     *                  -dpoly(mk,j)*s(mk,2,k)
c
      cv(mp,jj)= cv(mp,jj)+dpoly(ml,j)*s(ml,2,k)
     *                  -dpoly(mk,j)*s(mk,2,k)
c
   40 continue
c
      m1= m1+l
   30 continue
   20 continue
c
      call rfftmlt(cu,work,trigs,ifax,1,jump,im,jm,1)
      call rfftmlt(cv,work,trigs,ifax,1,jump,im,jm,1)
c
      do 22 j=1,jm
CDIR@ IVDEP
      do 22 i=1,im
      dlpl(i,j,k)= -cu(i,j)
      dtpl(i,j,k)= -cv(i,j)
   22 continue
c
   15 continue
      return
      end
      subroutine var(imjm,fac,x,xv,avsum)
c
      dimension x(imjm),xv(imjm),avsum(imjm)
c
      do 10 i=1,imjm
      avsum(i)= fac*xv(i)-(x(i)*fac)**2
   10 continue
      return
      end
      subroutine wndfix(x,imx,jmx)
c
c  programmer -- rosmond t. (02 nov 1989)
c
c  this routine computes values of wind field at poles and
c  de-weights winds with cos(lat)
c
c  ********************************************************************
```

```
c
c
      dimension x(imx,jmx)
      dimension cosl(200)
      save cosl,jmm,im2,pi,jmsav
c
      data jmsav/-100/
c
      if(jmx.ne.jmsav) then
      pi= 4.0*atan(1.0)
      do 3 j=1,jmx
      cosl(j)= pi*(0.5-(j-1.0)/(jmx-1.0))
      cosl(j)= cos(cosl(j))
      cosl(j)= 6.375e6/cosl(j)
    3 continue
      jmm= jmx-1
      im2= imx/2
      jmsav= jmx
      endif
c
      do 1 j=2,jmm
      do 1 i=1,imx
      x(i,j)= x(i,j)*cosl(j)
    1 continue
c
CDIR$ IVDEP
      do 5 i=1,im2
      x(i,1)= 0.5*(x(i,2)-x(i+im2,2))
      x(i+im2,1)= -x(i,1)
      x(i,jmx)= 0.5*(x(i,jmm)-x(i+im2,jmm))
      x(i+im2,jmx)= -x(i,jmx)
    5 continue
      return
      end
      subroutine womega(im,jm,lm,onocos,bsig,dasig,dbsig,pt
     *, dtpl,dlpl,rdiv,ut,vt,sd)
c
      dimension bsig(lm+1),ut(im*jm,lm),vt(im*jm,lm),onocos(im*jm)
     *, dlpl(im*jm),dtpl(im*jm),pten(im*jm),rdiv(im*jm,lm)
     *, dasig(lm),dbsig(lm),pt(im*jm)
c
      dimension g(im*jm,lm),sd(im*jm,lm)
c
      k= 1
      do 22 i=1,im*jm
      g(i,k)=ut(i,k)*dlpl(i)*onocos(i)+vt(i,k)*dtpl(i)
      pten(i)=g(i,k)+rdiv(i,k)*pt(i)
      pten(i)= -pten(i)*dbsig(k)-rdiv(i,k)*dasig(k)
      sd(i,k+1)= pten(i)
      sd(i,1)= 0.0
   22 continue
c
      do 2 k=2,lm-1
      do 2 i=1,im*jm
      g(i,k)= ut(i,k)*dlpl(i)*onocos(i)+vt(i,k)*dtpl(i)
      a= g(i,k)+rdiv(i,k)*pt(i)
      pten(i)= pten(i)-a*dbsig(k)-rdiv(i,k)*dasig(k)
      sd(i,k+1)= pten(i)
    2 continue
c
      k= lm
      do 24 i=1,im*jm
      g(i,k)= ut(i,k)*dlpl(i)*onocos(i)+vt(i,k)*dtpl(i)
      a= g(i,k)+rdiv(i,k)*pt(i)
      pten(i)= pten(i)-a*dbsig(k)-rdiv(i,k)*dasig(k)
   24 continue
c
```

```
c  vertical velocity
c
      do 3 k=2,lm
      do 3 i=1,im*jm
      sd(i,k)= sd(i,k)-bsig(k)*pten(i)
    3 continue
c
       do 450 k=1,lm
c
       if(k.lt.lm) then
c
        do 443 i=1,im*jm
        sd(i,k)=bsig(k+1)*(0.5*(g(i,k)+g(i,k+1))+pten(i))+sd(i,k+1)
  443   continue
c
       else
c
        do 441 i=1,im*jm
        sd(i,lm)= g(i,lm)+pten(i)
  441 continue
      endif
  450 continue
c
      return
      end
```

DISTRIBUTION LIST

OFFICE OF NAVAL RESEARCH
ATTN A WEINSTEIN CODE 1122
OCEAN SCIENCES DIVISION
ARLINGTON VA 22217-5660

OFFICE OF NAVAL RESEARCH
ATTN CODE 1122 MARINE MET
800 NORTH QUINCY ST
ARLINGTON VA 22217-5660

OFFICE OF NAVAL RESEARCH
ATTN CODE 1122 PHYS OCE
800 NORTH QUINCY ST
ARLINGTON VA 22217-5660

DEFENSE TECH INFO CENTER 2
CODE DTIC-FD DOC PROC DIV
BLDG 5 CAMERON STATION
ALEXANDRIA VA 22304-6145

COMMANDING OFFICER 12
ATTN CODE 5227 DOCS SEC
NAVRSCHLAB
WASHINGTON DC 20375-5320

COMMANDING OFFICER
ATTN CODE 1221 CLASSIF MGT
NAVRSCHLAB
WASHINGTON DC 20375-5320

NAVRSCHLAB
ATTN LIBRARY
JCSSC MS 39529-5004

NAVPGSCOL
ATTN CODE MR
MONTEREY CA 93943-5000

NAVPGSCOL
ATTN CODE OC
MONTEREY CA 93943-5000

AFGL OPI
HANSCOM AFB MA 01731

AFGWC DAPL
ATTN TECH LIBRARY
OFFUTT AFB NE 68113

AFGL/LY
ATTN MET OFFICER
HANSCOM AFB MA 01731

DIRECTOR NMC
NWS NOAA
WWB W32 RM 204
WASHINGTON DC 20233

NATIONAL WEATHER SERVICE
WORLD WEATHER BLDG RM 307
5200 AUTH ROAD
CAMP SPRINGS MD 20023

DIRECTOR
GEOPHYS FLUID DYNAMICS LAB
NOAA PRINCETON UNIVERSITY
PO BOX 308
PRINCETON NJ 08540

NASA GODDARD SPACE CENTER
LABORATORY FOR ATMOS SCI
GREENBELT MD 20771

LIBRARY AUSTRALIAN NUMERICAL
METEOROLOGY RESEARCH CENTER
PO BOX 5089A
MELBOURNE VICTORIA 3001
AUSTRALIA

LIBRARY BIBLIOTHEQUE
ATMOSPHERIC ENVIRON SERV
4905 RUE DUFFERIN STREET
DOWNSVIEW ONTARIO
CANADA M3H 5T4

EUROPEAN CENTRE FOR MEDIUM
RANGE WEATHER FORECASTS
SHINFIELD PARK READING
BERKSHIRE RG29&X ENGLAND